## Contents

1 CDN Basics 3  
1.1 CDN Basics 3  

2 Traffic Control Overview 7  
2.1 Traffic Control Overview 7  

3 Administrator’s Guide 77  
3.1 Administrator’s Guide 77  

4 Developer’s Guide 221  
4.1 Developer’s Guide 221  

5 APIs 307  
5.1 Traffic Ops API 307  

6 Tools 889  
6.1 Tools 889  

7 FAQ 933  
7.1 FAQ 933  

8 Indices and Tables 935  
8.1 Glossary 935  

Python Module Index 947  
Index 949
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:
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.

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.

---

# 3: This Response may Only be Cached for 86400 Seconds

```
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
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
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
```
CHAPTER 2

Traffic Control Overview

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

2.1 Traffic Control Overview

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

2.1.1 Introduction

Traffic Control is a 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 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 Vault is used as a separate, key/value store, allowing administrators to harden the Traffic Vault 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 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:
Apache Traffic Control Documentation, Release 4.0.0

- 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.demo1.mycdn.ciab.test/. When the LDNS is resolving this video.demo1.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 cache servers. The client then connects to port 80 (HTTP) or port 443 (HTTPS) on the cache server, and sends the Host: value. 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.

Chapter 2. Traffic Control Overview
# 8: Traffic Router Redirect to Edge-tier Cache Server

<table>
<thead>
<tr>
<th>HTTP/1.1 302 Found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location: <a href="http://edge.demo1.mycdn.ciab.test/">http://edge.demo1.mycdn.ciab.test/</a></td>
</tr>
<tr>
<td>Content-Length: 0</td>
</tr>
<tr>
<td>Date: Tue, 13 Jan 2015 20:01:41 GMT</td>
</tr>
</tbody>
</table>

In this case Traffic Router has access to more information when selecting a cache server because it has a full HTTP request instead of just a hostname. Traffic Router can be configured to select a cache server based on any of the following parts of the HTTP request:

- The client’s IP address.
- The URL the client is requesting.
- 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.

## 2.1.4 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 downstream 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 downstream 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

Optimistic 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 a downstream 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 downstream 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.

**Optimistic Quorum**

In order to prevent split-brain monitoring scenarios, a minimum of three Traffic Monitors are required to properly monitor a given CDN and the optimistic quorum feature should be enabled. If three or more Traffic Monitors are set to `ONLINE`, the optimistic quorum can be employed by setting the `peer_optimistic_quorum_min` property in `traffic_monitor.cfg` to a value greater than zero. This value represents the minimum number of peers that must be available in order to participate in the Optimistic Health Protocol. If Traffic Monitor detects that the number of available peers is less than this number, Traffic Monitor withdraws itself from participation in the health protocol by serving 503s for cache health state calls until connectivity is restored.

The optimistic quorum prevents invalid state propagation caused by a Traffic Monitor losing connectivity to the network and consequently marking all peers and caches as unavailable. When connectivity is restored, a race between peering recovery and polling from Traffic Routers begins. If Traffic Router were to poll a Traffic Monitor that has no available peers and optimistic quorum is not enabled or cannot be used (i.e.: too few Traffic Monitors), the Traffic Monitor will serve its local state only until peer connectivity is restored. If Traffic Router polls the Traffic Monitor when in this state, that is, prior to regaining peering, negative cache states caused by the lack of connectivity would be consumed and directly impact which caches are available for consideration for routing, until the Traffic Router polls a Traffic Monitor that has good state, or peering is restored. For this reason, it is recommended to run a minimum of three Traffic Monitors, with `peer_optimistic_quorum_min` set to a value of 1 or greater. Note that this value cannot exceed the number of peers of any given Traffic Monitor; that is, a value of 2 is the maximum value that can be used when three Traffic Monitors are in use. If this number exceeds the number of peers, the Traffic Monitor will always serve 503s and an error will be logged.
Protocol Engagement

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.

2.1.5 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

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 deliveryservice_stats are:
  - kbps
  - status_4xx
  - status_5xx
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
  - Bytes Served

Daily stats are stored by CDN.

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

### 2.1.6 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.

### 2.1.7 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 deliveryservices 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.

Consistent Hashing Regular Expression

When Traffic Router performs Consistent Hashing on a client request to find an Edge-tier cache server to which to redirect them, it can optionally first modify the request path by extracting the pieces that match this regular expression.

See also:
Consistent Hashing Patterns

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>consisten-tHashRegex</td>
<td>In source code and Traffic Ops API requests and responses</td>
<td>unchanged (regular expression)</td>
</tr>
<tr>
<td>pattern-based consistent hashing</td>
<td>documentation and the Traffic Portal UI</td>
<td>unchanged (regular expression), but usually used when discussing the concept rather than the field</td>
</tr>
</tbody>
</table>

Consistent Hashing Query Parameters

When Traffic Router performs Consistent Hashing on a client request to find an Edge-tier cache server to which to redirect them, it can optionally take into account any number of query parameters. This field defines them, formally as a Set but often represented as an Array/List due to encoding limitations. That is, if the Consistent Hashing Query Parameters on a Delivery Service are \{test\} and a client makes a request for /?test=something they will be directed to a different cache server than a different client that requests /?test=somethingElse, but the same cache server as a client that requests /?test=something&quest=somethingToo.

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>consistentHash-Query-Params</td>
<td>In source code, Traffic Portal, and Traffic Ops API requests and responses</td>
<td>unchanged (Array of strings - should ALWAYS be unique, thus treated as a Set in most contexts)</td>
</tr>
</tbody>
</table>
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</td>
<td>In Delivery Service objects returned by the Traffic Ops API</td>
<td>unchanged (int, integer etc.)</td>
</tr>
<tr>
<td>DNS 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.

**EDNS0 Client Subnet Enabled**

A boolean value that controls whether or not EDNS0 client subnet is enabled on this Delivery Service by Traffic Router. When creating a Delivery Service in Traffic Portal, this will default to “false”.

**Example URLs**

The Example URLs of a Delivery Service are the scheme/host specifications that clients can use to request content through it. These are determined by Traffic Ops from the Delivery Service’s configuration, and are read-only in virtually every context. The only reason a Delivery Service should ever have no Example URLs is if it is an ANY_MAP-Type Delivery Service (since they are not routed). For example, a Delivery Service that can deliver HTTP and HTTPS content, has a Routing Name of “cdn”, an xml_id of “demo1”, and belonging to a CDN that is authoritative for the mycdn.ciab.test domain would have two Example URLs:

- https://cdn.demo1.mycdn.ciab.test
- http://cdn.demo1.mycdn.ciab.test

Note that these are irrespective of request path; meaning a client can request e.g. https://cdn.demo1.mycdn.ciab.test/index.html through this Delivery Service.

**Warning:** This list does not consider any Static DNS Entries configured on the Delivery Service, those are

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delivery Service URLs</td>
<td>Traffic Portal forms</td>
<td>unchanged (list of strings)</td>
</tr>
</tbody>
</table>

2.1. Traffic Control Overview
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)

See also:

This is implemented using the ATS fq_pacing plign.

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FQRate</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 File. 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 Snapshot structures, especially in Traffic Ops API responses</td>
<td>A boolean which, if <code>true</code>, tells Traffic Router to only service requests when the client IP address is found in the Coverage Zone File</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>
<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 9: 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: <strong>INVALID_URL</strong> The Geo Limit Redirect URL has not yet been parsed, or an error occurred during parsing <strong>DS_URL</strong> The Geo Limit Redirect URL is served by this Delivery Service <strong>NOT_DS_URL</strong> 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-\textit{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: \textbf{MaxMind} IP address to geographic location mapping will be provided by a MaxMind GeoIP2 database.

1: \textbf{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>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>geo-Provider</td>
<td>Traffic Ops and Traffic Ops client code, \textit{Traffic Ops API} requests and responses</td>
<td>unchanged (integral, unique identifier)</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>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>missLat</td>
<td>In \textit{Traffic Ops API} 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 12: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>miss-Long</td>
<td>In <em>Traffic Ops API</em> responses and Traffic Ops source code</td>
<td>unchanged (numeric)</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 13: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>totalKbpsThreshold</td>
<td>In <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 14: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>totalTpsThreshold</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”.

Chapter 2. Traffic Control Overview
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 Traffic Ops API 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>longDesc</td>
<td>Traffic Control source code and Traffic Ops API responses</td>
<td>unchanged (string, String etc.)</td>
</tr>
</tbody>
</table>

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.
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>longDesc2</td>
<td>Traffic Control source code and Traffic Ops API 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:

**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”

\[\text{Note: This regular expression type can only exist in the Match List of STEERING-Type Delivery Services - and not CLIENT_STEERING.}\]

### Table 15: 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 Ops</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

\(^4\) These regular expression types can only appear in the Match List of HTTP-Routed Delivery Services.
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.

**Max Origin Connections**

The maximum number of TCP connections individual *Mid-tier cache servers* are allowed to make to the *Origin Server Base URL*. A value of 0 in this field indicates that there is no maximum.

**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`. Must not include paths, query parameters, document fragment identifiers, or username/password URL fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>orgServer-Fqdn</td>
<td><em>Traffic Ops API</em> 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 the *ID* of said *Profile*.
### Table 17: 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>profile-Name</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 18: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>CDN Snapshots</td>
<td>An object containing the key &quot;acceptHttps&quot; that is also a string containing a boolean which expresses whether or not 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>
</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 pass up” in Traffic Portal forms)

2  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 19: 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, <em>Traffic Ops API</em> 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 its 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)

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)

**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.
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.

**See also:**

The Apache Trafficserver documentation for the Regex Remap plugin

<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 <em>Traffic Ops API</em> 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 regu-
lar expression matches a client’s request path (i.e. not a full URL - /path/to/content 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.

See also:
The Apache Trafficserver documentation for the Regex Remap plugin

<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</td>
</tr>
<tr>
<td></td>
<td>requests/responses</td>
<td>(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.
Required Capabilities

New in version ATCv4.

A Delivery Service can be associated with Server Capabilities that it requires cache servers serving its content to have. When one or more Server Capability is required by a Delivery Service, it will block the assignment of cache servers to it that do not have those Server Capabilities. Additionally, the Edge-tier Cache Servers assigned to a Delivery Service that requires a Server Capability will only request content they do not have cached from Mid-tier Cache Servers which also have this Server Capability.

Typically, a required Server Capability is represented merely by the name of said Server Capability. In fact, there’s nothing more to a Server Capability than its name; it’s the responsibility of CDN operators to ensure that they are assigned and required properly. There is no mechanism to detect whether or not a cache server has a given Server Capability, it must be assigned manually.

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\(^1\).

Servers

Servers can be assigned to Delivery Services using the Servers and Delivery Services 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.

\(^1\) 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.
See also:

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.

**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>
<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>

Keys for either algorithm can be generated within Traffic Portal.

**SSL Key Version**

An integer that describes the version of the SSL key(s) - if any - used by this Delivery Service. This is incremented whenever Traffic Portal generates new SSL keys for the Delivery Service.

**Warning:** This number will not be correct if keys are manually replaced using the API, as the key generation API does not increment it!

**Static DNS Entries**

Static DNS Entries can be added under a Delivery Service’s domain. These DNS records can be configured in the Delivery Services 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 Traffic Ops API 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>trResponse-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>

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>
<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
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:

- **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** 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.

---

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.
HTTP_LIVE³ 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_CACHE³ 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.

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 sub-
sequent 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 26: 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 27: 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 28: 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>UN- KNOWN - 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></td>
<td>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 29: Aliases

<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>
2.1.9 Cache Groups

A Cache Group is - ostensibly - exactly what it sounds like it is: a group of cache servers. More specifically, every server in a Traffic Control CDN must be in a Cache Group (even if they are not actually cache servers). Typically a Cache Group is representative of the available cache servers within a specific geographical location. Despite that cache servers have their own Physical Locations, when cache servers are chosen to serve content to a client based on geographic location the geographic location actually used for comparisons is that for the Cache Group that contains it, not the geographic location of the cache server itself.

The most typical Types of Cache Groups are EDGE_LOC which contain Edge-tier cache servers and MID_LOC which contain Mid-tier cache servers. The latter are each designated as a Parent of one or more of the former to fill out the two-tiered caching hierarchy of an ATC CDN.

Consider the example CDN in Fig. 1. 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. 1: An example CDN that shows the hierarchy between four Edge-tier Cache Groups, two Mid-tier Cache Groups, and one Origin

Regions, Divisions, and Locations

In addition to being in a Cache Group, 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 Topography Hierarchy.

To create these structures in Traffic Portal, first make at least one Division under Topology → 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).
Fig. 2: Topography Hierarchy
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. There is no strict requirement that *cache servers* in a Cache Group share a *Physical Location*, *Region*, or *Division*. This may be confusing at first as there are a few places in code, interfaces, or even documentation where Cache Groups are referred to as “Cache Locations” or even erroneously as “Physical Locations”.

### Properties

Cache Groups 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 Cache Group 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 Cache Group-related endpoints such as `cachegroups` contains definitions of the Cache Group object(s) returned and/or accepted by those endpoints.

### ASN

A Cache group can have zero or more ASNs (Autonomous System Numbers) assigned to it, which is used to classify traffic that passes through a CDN. These are typically not represented on a Cache Group object itself, but rather as a separate object indicating the relationship, e.g. in the requests and responses of the `asns` endpoint.

**See also:**

The [Autonomous System Wikipedia page](https://en.wikipedia.org/wiki/Autonomous_System) for an explanation of what an ASN (Autonomous System Number) actually is.

### Coordinate

**Tip:** Normally, one need not interact with this. In most contexts, this property of a Cache Group is not even exposed, but instead the Cache Group’s *Latitude* and *Longitude* are exposed and should be directly manipulated.
The *Coordinate* of a Cache Group defines the geographic coordinates of a Cache Group that is used for routing clients based on geographic location. It is also used to determine the “closest” Cache Group to another for the purposes of *Fallback to Closest*.

Typically, this is expressed as an integral, unique identifier for the “Coordinate” object bound to the Cache Group that defines its geographic location, but occasionally it may appear as the name of that “Coordinate” object.

**Note:** When a new Cache Group is created, it is not necessary to first create a “Coordinate” object where it may reside. Instead, “Coordinates” are created automatically to reflect the *Latitude* and *Longitude* given to the newly created Cache Group. The name of the generated “Coordinate” will conform to the pattern `from_cachegroup_Name` where *Name* is the Cache Group’s *Name*. Because of this, creating new Cache Groups will fail if a “Coordinate” with a name matching that pattern already exists.

### Fallbacks

*Fallbacks* are a group of zero or more Cache Groups to be considered for routing when a Cache Group becomes unavailable due to high load or excessive maintenance. These are normally represented by an array of each Cache Group’s *ID*, but may occasionally appear as the *Name* or *Short Name* of each Cache Group.

This set is consulted before *Fallback to Closest* is taken into consideration.

**See also:**

*Health Protocol*

**Table 30: Aliases**

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failover Cache Groups</td>
<td>Traffic Portal forms - but <strong>not</strong> tables</td>
<td>List or array of <em>Names</em> as strings</td>
</tr>
<tr>
<td>backupLocations</td>
<td>In CDN <em>Snapshots</em></td>
<td>A sub-object called “list” which is a list or array of Cache Group <em>Names</em> as strings</td>
</tr>
<tr>
<td>Backup-CacheGroups</td>
<td>Traffic Router source code</td>
<td>A List of strings that are the <em>Names</em> of Cache Groups</td>
</tr>
</tbody>
</table>

### Fallback to Closest

This is a boolean field which, when `true` (True, TRUE etc.) causes routing to “fall back” on the nearest Cache Group - geographically - when this Cache Group becomes unavailable due to high load and/or excessive maintenance.

When this *is* a “true” value, the closest Cache Group will be chosen if and only if any set of *Fallbacks* configured on the Cache Group has already been exhausted and no available Cache Groups were found..
See also:

*Health Protocol*

Table 31: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fallback to Geo Failover</td>
<td>Traffic Portal forms</td>
<td>Unchanged (bool, Boolean etc.)</td>
</tr>
</tbody>
</table>

**ID**

All Cache Groups have an integral, unique identifier that is mainly used to reference it in the *Traffic Ops API*.

Despite that a Cache Group’s *Name* must be unique, this is the identifier most commonly used to represent a unique Cache Group in most contexts throughout ATC. One notable exception is in CDN *Snapshots* and in routing configuration used by Traffic Router.

**Latitude**

The Cache Group’s geomagnetic latitude for use in routing and for the purposes of *Fallback to Closest*.

Table 32: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geo Magnetic Latitude</td>
<td>Traffic Portal forms</td>
<td>Unchanged (number, float etc.)</td>
</tr>
</tbody>
</table>

**Localization Methods**

The *Localization Methods* of a Cache Group define the methods by which Traffic Router is allowed to route clients to *cache servers* within this Cache Group. This is a collection of the allowed methods, and the values in the collection are restricted to the following.

- “Coverage Zone File” (alias `CZ` in source code, database entries, and *Traffic Ops API* requests/responses) allows Traffic Router to direct clients to this Cache Group if they were assigned a geographic location by looking up their IP address in the *Coverage Zone File*.

- “Deep Coverage Zone File” (alias `DEEP_CZ` in source code, database entries, and *Traffic Ops API* requests/responses) was intended to allow Traffic Router to direct clients to this Cache Group if they were assigned a geographic location by looking up their IP addresses in the *Deep Coverage Zone File*. However, it **has no effect at all**. This option therefore will not appear in Traffic Portal forms.
Warning: In order to make use of “deep caching” for a Delivery Service, all that is required is that Delivery Service has Deep Caching enabled. If that is done and a cache server appears in the Deep Coverage Zone File then clients can and will be routed using that method. There is no way to disable this behavior on a Cache Group (or otherwise) basis, and the presence or absence of DEEP_CZ in a Cache Group’s Localization Methods has no meaning.

- “Geo-IP Database” (alias GEO in source code, database entries, and Traffic Ops API requests/responses) allows Traffic Router direct clients to this Cache Group if the client’s IP was looked up in a provided IP address-to-geographic location mapping database to provide their geographic location.

If none of these localization methods are in the set of allowed methods on a Cache Group, it is assumed that clients should be allowed to be routed to that Cache Group regardless of the method used to determine their geographic location.

This property only has meaning for Cache Groups containing Edge-tier cache servers. Which is to say (one would hope) that it only has meaning for EDGE_LOC Cache Groups.

### Table 33: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled Localization Methods</td>
<td>Traffic Portal forms, Traffic Router source code</td>
<td>Unchanged (Set&lt;String&gt;, Array&lt;string&gt; etc.)</td>
</tr>
</tbody>
</table>

### Longitude

The Cache Group’s geomagnetic longitude for use in routing and for the purposes of Fallback to Closest.

### Table 34: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geo Magnetic Longitude</td>
<td>Traffic Portal forms</td>
<td>Unchanged (number, float etc.)</td>
</tr>
</tbody>
</table>

### Name

A unique, human-friendly name for the Cache Group, with no special character restrictions or length limit.

Though this property must be unique for all Cache Groups, ID is more commonly used as a unique identifier for a Cache Group in most contexts.
### Table 35: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>locationID</td>
<td>CDN Snapshots</td>
<td>Unchanged (str, String etc.)</td>
</tr>
</tbody>
</table>

### Parent

A Cache Group can have a `parent` Cache Group which has different meanings based on the Cache Group’s `Type`.

- An `EDGE_LOC` Cache Group’s parent must be a `MID_LOC` Cache Group. When configuration files are generated for Edge-tier cache servers, their parents (different from a Cache Group parent) will be selected from the Mid-tier cache servers contained within the Cache Group that is the parent of their containing Cache Group.

- A `MID_LOC` Cache Group’s parent must be an `ORG_LOC` Cache Group. However, if any given `MID_LOC` either doesn’t have a parent, or does and it’s an `ORG_LOC`, then all `ORG_LOC` Cache Groups - even across CDNs - will be considered the parents of that `MID_LOC` Cache Group. This parent relationship only has any meaning in the context of “multi-site-origins”, as they are unnecessary in other scenarios.

**See also:**

*Configure Multi-Site Origin*

- For all other Cache Group `Types`, parent relationships have no meaningful semantics.

**Danger:** There is no safeguard in the data model or *Traffic Ops API* that ensures these relationships hold. If they are violated, the resulting CDN behavior is undefined - and almost certainly undesirable.

*Parents* are typically represented by their `ID`, but may occasionally appear as their `Name`.

**See also:**

The Apache Traffic Server documentation for the “parent.config” configuration file. ATC parentage relationships boil down to the `parent` field of that configuration file, which sets the `parents` of cache servers.

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent Cache Group</td>
<td>Traffic Portal forms and tables</td>
<td>Unchanged (str, String etc.)</td>
</tr>
<tr>
<td>Parent-CacheGroupID</td>
<td>Traffic Ops database, Traffic Ops Go code, <em>Traffic Ops API</em> requests and responses</td>
<td>Positive integer (int, bigint etc.)</td>
</tr>
<tr>
<td>Parent-CacheGroupName</td>
<td>Traffic Ops database, Traffic Ops Go code, <em>Traffic Ops API</em> requests and responses</td>
<td>Unchanged (str, String etc.)</td>
</tr>
</tbody>
</table>
Parameters

For unknown reasons, it’s possible to assign Parameters to Cache Groups. This has no attached semantics respected by ATC, and exists today only for compatibility purposes.

These are nearly always represented as a collection of Parameter IDs but occasionally they can be expressed as full Parameter objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cachegroup-Parameters</td>
<td>Certain Traffic Ops API responses, sometimes in internal Go code</td>
<td>Usually a tuple of associative information like a Cache Group ID and a Parameter ID</td>
</tr>
</tbody>
</table>

Secondary Parent

A secondary parent of a Cache Group is used for fall-back purposes on a cache server-to-cache server basis after routing has already occurred (in contrast with Fallbacks and Fallback to Closest which operate at the routing step on a Cache Group-to-Cache Group basis).

For an explanation of what it means for one Cache Group to be the Parent of another, refer to the Parent section.

See also:

The Apache Traffic Server documentation for the “parent.config” configuration file. ATC secondary parentage relationships boil down to the secondary_parent field of that configuration file, which sets the “secondary parents” of cache servers.

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secondary Parent Cache Group</td>
<td>Traffic Portal forms and tables</td>
<td>Unchanged (str, String etc.)</td>
</tr>
<tr>
<td>SecondaryParent-CacheGroupID</td>
<td>Traffic Ops database, Traffic Ops Go code, Traffic Ops API requests and responses</td>
<td>Positive integer (int, bigint etc.)</td>
</tr>
<tr>
<td>SecondaryParent-CacheGroupName</td>
<td>Traffic Ops database, Traffic Ops Go code, Traffic Ops API requests and responses</td>
<td>Unchanged (str, String etc.)</td>
</tr>
</tbody>
</table>

Servers

The primary purpose of a Cache Group is to contain servers. In most scenarios it is implied or assumed that the servers are cache servers, but this is not required and in fact it is not by any means uncommon for the contained servers to be of some other arbitrary Type.

A Cache Group can have zero or more assigned servers, and each server can belong to at most one Cache Group.
Short Name

This is typically an abbreviation of the Cache Group’s Name. The main difference is that it isn’t required to be unique.

Type

A Cache Group’s Type determines what kind of servers it contains. Note that there’s no real restriction on the kinds of servers that a Cache Group can contain, and this Type serves as more of a guide in certain contexts. The Types available by default are described in this section.

Tip: Because Types are mutable, the actual Types that can describe Cache Groups cannot be completely or precisely defined. However, there are no good reasons of which this author can think to modify or delete the default Types herein described, and honestly the good reasons to even merely add to their ranks are likely few.

Table 39: Aliases

<table>
<thead>
<tr>
<th>Name</th>
<th>Use(s)</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Traffic Ops client and server Go code, Traffic Ops API requests and responses</td>
<td>positive integer (int, bigint, etc.)</td>
</tr>
<tr>
<td>ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Traffic Ops API requests and responses, Traffic Ops database</td>
<td>unchanged (string, String etc.)</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

EDGE_LOC

This Type of Cache Group contains Edge-tier cache servers.

MID_LOC

This Type of Cache Group contains Mid-tier cache servers.

ORG_LOC

This Type of Cache Group contains origins. The primary purpose of these is to group origins for the purposes of “multi-site-origins”, and it’s suggested that if that doesn’t meet your use-case that these be mostly avoided. In general, it’s not strictly necessary to create origin servers in ATC at all, unless you have to support “multi-site-origins”.

See also:

Configure Multi-Site Origin
A catch-all *Type* of Cache Group that’s meant to house infrastructure servers that gain no special semantics based on the Cache Group containing them, e.g. Traffic Portal instances.

This *Type* of Cache Group is meant specifically to contain Traffic Router instances.

### 2.1.10 Profiles and Parameters

*Profiles* are a collection of configuration options, defined partially by the Profile’s *Type* (not to be confused with the more general “*Type*” used by many other things in Traffic Control) and partially by the *Parameters* set on them. Mainly, Profiles and Parameters are used to configure *cache servers*, but they can also be used to configure parts of (nearly) any Traffic Control component, and can even be linked with more abstract concepts like *Delivery Services* and *Cache Groups*. The vast majority of configuration done within a Traffic Control CDN must be done through *Profiles* and *Parameters*, which can be achieved either through the *Traffic Ops API* or in the *Profiles* view of Traffic Portal. For ease of use, Traffic Portal allows for duplication, comparison, import and export of *Profiles* including all of their associated *Parameters*.

#### Profiles

##### Properties

Profile objects as represented in e.g. the *Traffic Ops API* or in the *Traffic Portal Profiles view* have several properties that describe their general operation. In certain contexts, the *Parameters* assigned to a Profile (and/or the integral, unique identifiers thereof) may appear as properties of the Profile, but that will not appear in this section as a description of *Parameters* is provided in the section of that name.

##### CDN

A Profile is restricted to operate within a single CDN. Often, “CDN” (or “cdn”) refers to the integral, unique identifier of the CDN, but occasionally it refers to the *name* of said CDN. It may also appear as e.g. *cdnId* or *cdnName* in *Traffic Ops API* payloads and responses. A Profile may only be assigned to a server, *Delivery Service*, or *Cache Group* within the same CDN as the Profile itself.

#### Description

Profiles may have a description provided by the creating user (or Traffic Control itself in the case of the *Default Profiles*). The *Traffic Ops API* does not enforce length requirements on
the description (though Traffic Portal does), and so it’s possible for Profiles to have empty
descriptions, though it is strongly recommended that Profiles have meaningful descriptions.

ID

An integral, unique identifier for the Profile.

Name

Ostensibly this is simply the Profile’s name. However, the name of a Profile has drastic conse-
quences for how Traffic Control treats it. Particularly, the name of a Profile is heavily conflated
with its Type. These relationships are discussed further in the Type section, on a Type-by-Type
basis.

Routing Disabled

This property can - and in fact must - exist on a Profile of any Type, but it only has any meaning
on a Profile that has a name matching the constraints placed on the names of ATS_PROFILE-
Type Profiles. This means that it will also have meaning on Profiles of Type UNK_PROFILE
that for whatever reason have names beginning with EDGE or MID. When this field is defined
as 1 (may be displayed as true in e.g. Traffic Portal), Traffic Router will not be informed of
any Delivery Services to which the cache server using this Profile may be assigned. Effectively,
this means that client traffic cannot be routed to them, although existing connections would be
uninterrupted.

Type

A Profile’s Type determines how its configured Parameters are treated by various components,
and often even determine how the object using the Profile is treated (particularly when it is a
server). Unlike the more general “Type” employed by Traffic Control, the allowed Types of
Profiles are set in stone, and they are as follows.

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.

ATSPROFILE A Profile that can be used with either an Edge-tier or Mid-tier cache server
(but not both, in general). This is the only Profile type that will ultimately pass its Parameters
on to ORT in the form of generated configuration files. For this reason, it can make
use of the Strings with Special Meaning to ORT in the values of some of its Parameters.
**DS_PROFILE**  A Profile that, rather than applying to a server, is instead used by a Delivery Service.

**ES_PROFILE**  A Profile for ElasticSearch servers. This has no known special meaning to any component of Traffic Control, but if ElasticSearch is in use the use of this Profile Type is suggested regardless.

**GROVE_PROFILE**  A Profile for use with the experimental Grove HTTP caching proxy.

**INFLUXDB_PROFILE**  A Profile used with InfluxDB, which is used by Traffic Stats.

**KAFKA_PROFILE**  A Profile for Kafka servers. This has no known special meaning to any component of Traffic Control, but if Kafka is in use the use of this Profile Type is suggested regardless.

**LOGSTASH_PROFILE**  A Profile for Logstash servers. This has no known special meaning to any component of Traffic Control, but if Logstash is in use the use of this Profile Type is suggested regardless.

**ORG_PROFILE**  A Profile that may be used by either origin servers or origins (no, they aren’t
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!

**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. This has no known special meaning to any component of Traffic Control, but if Splunk is in use the use of this Profile Type is suggested regardless.

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!

**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!

**TP_PROFILE** A Traffic Portal Profile. This has no known special meaning to any Traffic Control component(s) (not even Traffic Portal itself), but its use is suggested for the profiles used by any and all Traffic Portal servers anyway.

**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!

See also:

*The Traffic Router Profile*
TS_PROFILE A Traffic Stats Profile.

Caution: For legacy reasons, the names of Profiles of this type must be 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! Furthermore, because Profile names must be unique, this means that only one TS_PROFILE-Type Profile can exist at a time.

UNK_PROFILE A catch-all type that can be assigned to anything without imbuing it with any special meaning or behavior.

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.

Default Profiles

Traffic Control comes with some pre-installed Profiles for its basic components, but users are free to define their own as needed. Additionally, these default Profiles can be modified or even removed completely. One of these Profiles is The GLOBAL Profile, which has a dedicated section.

INFLUXDB A Profile used by InfluxDB servers that store Traffic Stats information. It has a Type of UNK_PROFILE and is assigned to the special “ALL” CDN.

RIAK_ALL This Profile is used by Traffic Vault, which is, generally speaking, the only instance in Traffic Control as it can store keys for an arbitrary number of CDNs. It has a Type of UNK_PROFILE and is assigned to the special “ALL” CDN.

TRAFFIC_ANALYTICS A default Profile that was intended for use with the now-unplanned “Traffic Analytics” ATC component. It has a Type of UNK_PROFILE and is assigned to the special “ALL” CDN.

TRAFFIC_OPS A Profile used by the Traffic Ops server itself. It’s suggested that any and all “mirrors” of Traffic Ops for a given Traffic Control instance be recorded separately and all assigned to this Profile for record-keeping purposes. It has a Type of UNK_PROFILE and is assigned to the special “ALL” CDN.

TRAFFIC_OPS_DB A Profile used by the PostgreSQL database server that stores all of the data needed by Traffic Ops. It has a Type of UNK_PROFILE and is assigned to the special “ALL” CDN.

TRAFFIC_PORTAL A Profile used by Traffic Portal servers. This profile name has no known special meaning to any Traffic Control components (not even Traffic Portal itself), but its use is suggested for Traffic Portal servers anyway. It has a Type of UNK_PROFILE and is assigned to the special “ALL” CDN.

TRAFFIC_STATS This is the only Profile used by Traffic Stats (though InfluxDB servers have their own Profile(s)). It has a Type of UNK_PROFILE and is assigned to the special
“ALL” CDN.

In addition to these Profiles, each release of Apache Traffic Control is accompanied by a set of suggested Profiles suitable for import in the Profiles view of Traffic Portal. They may be found on the Profiles Downloads Index page. These Profiles are typically built from production Profiles by a company using Traffic Control, and as such are typically highly specific to the hardware and network infrastructure available to them. **None of the Profiles bundled with a release are suitable for immediate use without modification**, and in fact many of them cannot actually be imported directly into a new Traffic Control environment, because Profiles with the same Names already exist (as above).

Administrators may alternatively wish to consult the Profiles and Parameters available in the CDN in a Box environment, as they might be more familiar with them. Furthermore, those Profiles are built with a minimum running Traffic Control system in mind, and thus may be easier to look through. The Profiles and their associated Parameters may be found within the infrastructure/cdn-in-a-box/traffic_ops_data/profiles/ directory.

**The GLOBAL Profile**

There is a special Profile of Type UNK_PROFILE that holds global configuration information - its Name is “GLOBAL”, its Type is UNK_PROFILE and it is assigned to the special “ALL” CDN. The Parameters that may be configured on this Profile are laid out in the Global Profile Parameters Table.
## Table 40: 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 generated configuration files.</td>
</tr>
<tr>
<td>tm.inforurl</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.logourl</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. Setting this Parameter can significantly lighten the load on the Traffic Monitor system and it is therefore recommended that this be set on a production system.</td>
</tr>
<tr>
<td>use_reval_pending</td>
<td>global</td>
<td>This Parameter is present and its Value is exactly “1”, Traffic Ops will separately keep track of cache servers’ updates and pending content invalidation jobs. This behavior should be enabled by default, and disabling it, while still possible, is EXTREMELY DISCOURAGED.</td>
</tr>
<tr>
<td>use_tenancy</td>
<td>global</td>
<td>This Parameter, when it exists and has a Value of exactly “1” enables the use of Tenants in Traffic Control. This should be enabled by default, and while disabling this is still possible, it is EXTREMELY DISCOURAGED.</td>
</tr>
<tr>
<td>geolocation.polling.url</td>
<td>CR-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>CR-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>CR-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 the default geographic IP mapping database used by Traffic Control is MaxMind’s GeoIP2 database. The format of this Parameter’s Value is: Country Code;Latitude,Longitude, e.g. US;37.751,-97.822</td>
</tr>
<tr>
<td>regex_revalidation.Duration</td>
<td>regex_revalidate.config</td>
<td>This Parameter sets the maximum duration, in days, for which a content invalidation job may run. This is extremely important, as there is currently no way to delete a content invalidation job once it has been created. Furthermore, while there is no restriction placed on creating multiple Parameters with this Name, it is extremely discouraged as the actual maximum duration for content invalidation jobs is undefined, and can differ from server to server, and configuration file to configuration file.</td>
</tr>
</tbody>
</table>
Note: Since the Traffic Ops UI has been removed, the tm.logourl has no real meaning, and in fact most Traffic Ops distributions neither set this Parameter, nor provide a logo.

Some of these Parameters have the Config File value global, while others have CRConfig.json. This is not a typo, and the distinction is that those that use global are typically configuration options relating to Traffic Control as a whole or to Traffic Ops itself, whereas CRConfig.json is used by configuration options that are set globally, but pertain mainly to routing and are thus communicated to Traffic Routers through CDN Snapshots (which historically were called “CRConfig Snapshots” or simply “the CRConfig”). When a Parameter has a Config File value that isn’t one of global or CRConfig.json, it refers to the global configuration of said Config File across all servers that use it across all CDNs configured in Traffic Control. This can be used to easily apply extremely common configuration to a great many servers in one place.

Parameters

A Parameter is usually a way to set a line in a configuration file that will appear on the servers using Profiles that have said Parameter. More generally, though, a Parameter merely describes some kind of configuration for some aspect of some thing. There are many Parameters that must exist for Traffic Control to work properly, such as those on The GLOBAL Profile or the Default Profiles. Some Traffic Control components can be associated with Profiles that only have a few allowed (or actually just meaningful - others are ignored and don’t cause problems) but some can have any number of Parameters to describe custom configuration of things of which Traffic Control itself may not even be aware (most notably cache servers). For most Parameters, the meaning of each Parameter’s various properties are very heavily tied to the allowed contents of Apache Traffic Server configuration files.

Properties

When represented in Traffic Portal (in the Parameters view) or in Traffic Ops API request and/or response payloads, a Parameter has several properties that define it. In some of these contexts, the Profiles to which a Parameter is assigned (and/or the integral, unique identifiers thereof) are represented as a property of the Parameter. However, an explanation of this “property” is not provided here, as the Profiles section exists for the purpose of explaining those.

Config File

This (usually) names the configuration file to which the Parameter belongs. Note that it is only the name of the file and not the full path to the file - e.g. remap.config not /opt/trafficserver/etc/trafficserver/remap.config. To define the full path to any given configuration file, Traffic Ops relies on a reserved Name value: “location”.

See also:

This section is only meant to cover the special handling of Parameters assigned to specific Config File values. It is not meant to be a primer on Apache Traffic Server configuration files,
nor is it intended to be exhaustive of the manners in which said files may be manipulated by Traffic Control. For more information, consult the documentation for Apache Traffic Server configuration files.

Certain Config Files are handled specially by Traffic Ops’s configuration file generation. Specifically, the format of the configuration is tailored to be correct when the syntax of a configuration file is known. However, these configuration files must have “location” Parameters on the Profile of servers, or they will not be generated. The Config File values that are special in this way are detailed within this section. When a Config File is none of these special values, each Parameter assigned to given server’s Profile with the same Config File value will create a single line in the resulting configuration file (with the possible exception being when the Name is “header”)

12M_facts

This legacy file is generated entirely from a Profile’s metadata, and cannot be affected by Parameters.

Tip: This Config File serves an unknown and likely historical purpose, so most users/administrators/developers don’t need to worry about it.

50-ats.rules

Parameters have no meaning when assigned to this Config File (except “location”), but it is affected by Parameters that are on the same Profile with the Config File storage.config. Not this Config File. For each letter in the special “Drive Letters” Parameter, a line will be added of the form KERNEL=="PrefixLetter", OWNER="ats" where Prefix is the Value of the Parameter with the Name “Drive Prefix” and the Config File storage.config - but with the first instance of /dev/ removed -, and Letter is the drive letter. Also, if the Parameter with the Name “RAM Drive Prefix” exists on the same Profile assigned to the server, a line will be inserted for each letter in the special “RAM Drive Letters” Parameter of the form KERNEL=="PrefixLetter", OWNER="ats" where Prefix is the Value of the “RAM Drive Prefix” Parameter - but with the first instance of /dev/ removed -, and Letter is the drive letter.

Tip: This Config File serves an unknown and likely historical purpose, so most users/administrators/developers don’t need to worry about it.

astats.config

This configuration file will be generated with a line for each Parameter with this Config File value on the cache server’s Profile in the form Name=Value where Name is the Parameter’s Name with trailing characters that match __\d+$ stripped, and Value is its Value.
bg_fetch.config

This configuration file always generates static contents besides the header, and cannot be affected by any Parameters (besides its “location” Parameter).

See also:

For an explanation of the contents of this file, consult the Background Fetch Apache Traffic Server plugin’s official documentation.

cache.config

This configuration is built entirely from Delivery Service configuration, and cannot be affected by Parameters.

See also:

The Apache Traffic Server cache.config documentation

cacheurlanything.config

Config Files that match this pattern - where anything is a string of zero or more characters - can only be generated by providing a location and their contents will be fully determined by properties of Delivery Services.

See also:

The official documentation for the Cache URL Apache Traffic Server plugin.

Deprecation since version ATCv3.0: This configuration file is only used by Apache Traffic Server version 6.x, whose use is deprecated both by that project and Traffic Control. These Config Files will have no special meaning at some point in the future.

chkconfig

This actually isn’t a configuration file at all, kind of. Specifically, it is a valid configuration file for the legacy chkconfig utility - but it is never written to disk on any cache server. Though all Traffic Control-supported systems are now using systemd(8), ORT still uses chkconfig-style configuration to set the status of services on its host system(s). This means that any Parameter with this Config File value should have a Name that is the name of a service on the cache servers using the Profile to which the Parameter is assigned, and it’s Value should be a valid chkconfig configuration line for that service.

CRConfig.json

In general, the term “CRConfig” refers to CDN Snapshots, which historically were called “CR-Config Snapshots” or simply “the CRConfig”. Parameters with this Config File should be only
be on either *The GLOBAL Profile* where they will affect global routing configuration, or on a Traffic Router’s *Profile* where they will affect routing configuration for that Traffic Router only.

**See also:**

For the available configuration Parameters for a Traffic Router Profile, see *The Traffic Router Profile*.

**drop_qstring.config**

This configuration file will be generated with a single line that is exactly: /([^?]+) $s:/ /$t/$1n unless a Parameter exists on the *Profile* with this Config File value, and the Name “content”. In the latter case, the contents of the file will be exactly the Parameter’s Value (with terminating newline appended).

**global**

In general, this Config File isn’t actually handled specially by Traffic Ops when generating server configuration files. However, this is the Config File value typically used for Parameters assigned to *The GLOBAL Profile* for truly “global” configuration options, and it is suggested that this precedent be maintained - i.e. don’t create Parameters with this Config File.

**hdr_rw_anything.config**

Config Files that match this pattern - where *anything* is zero or more characters - are written specially by Traffic Ops to accommodate the *DSCP* setting of *Delivery Services*.

**Tip:** The *anything* in those file names is typically a *Delivery Service’s xml_id* - though the inability to affect the file’s contents is utterly independent of whether or not a *Delivery Service* with that *xml_id* actually exists.

**See also:**

For information on the contents of files like this, consult the Header Rewrite Apache Traffic Server plugin’s documentation

**hosting.config**

This configuration file is mainly generated based on the assignments of *cache servers* to *Delivery Services* and the *Cache Group* hierarchy, but there are a couple of Parameter *Names* that can affect it when assigned to this Config File. When a Parameter assigned to the *storage* config Config File - **NOT this Config File** - with the Name “RAM_Drive_Prefix” *exists*, it will cause lines to be generated in this configuration file for each *Delivery Service* that is of on of the *Types* DNS_LIVE (only if the server is an *Edge-Tier Cache Server*), HTTP_LIVE (only if the server is an *Edge-Tier Cache Server*), DNS_LIVE_NATNL, or HTTP_LIVE_NATNL
to which the cache server to which the Profile containing that Parameter belongs is assigned. Specifically, it will cause each of them to use volume=1 UNLESS the Parameter with the Name “Drive_Prefix” associated with Config File storage.config - again, NOT this Config File - also exists, in which case they will use volume=2.

Caution: If a Parameter with Config File storage.config and Name “RAM_Drive_Prefix” does not exist on a Profile, then the cache servers using that Profile will be incapable of serving traffic for Delivery Services of the aforementioned Types, even when a “location” Parameter exists.

See also:
For an explanation of the syntax of this configuration file, refer to the Apache Traffic Server hosting.config documentation.

ip_allow.config

This configuration file is mostly generated from various server data, but can be affected by a Parameter that has a Name of “purge_allow_ip”, which will cause the insertion of a line with src_ip=VALUE action=ip_allow method=ALL where VALUE is the Parameter’s Value. Additionally, Parameters with Names like coalesce_masklen|number_v4|6 cause Traffic Ops to generate coalesced IP ranges in different ways. In the case that number was used, the Parameter’s Value sets the the maximum number of IP address that may be coalesced into a single range. If masklen was used, the lines that are generated are coalesced into CIDR (Classless Inter-Domain Routing) ranges using mask lengths determined by the Value of the parameter (using ‘4’ sets the mask length of IPv4 address coalescing while using ‘6’ sets the mask length to use when coalescing IPv6 addresses). This is not recommended, as the default mask lengths allow for maximum coalescence. Furthermore, if two Parameters on the same Profile assigned to a server having Config File values of ip_allow.config and Names that are both “coalesce_masklen_v4” but each has a different Value, then the actual mask length used to coalesce IPv4 addresses is undefined (but will be one of the two). All forms of the “coalescence Parameters” have this problem.

Implementation Detail

At the time of this writing, coalescence is implemented through the the NetAddr::IP Perl library.

See also:
The Apache Traffic Server ip_allow.config documentation explains the syntax and meaning of lines in that file.
logging.config

This configuration file can only be affected by Parameters with specific Names. Specifically, for each Parameter assigned to this Config File on the Profile used by the cache server with the name LogFormatN.Name where N is either the empty string or a natural number on the interval [1,9] the text in Log Format Snippet will be inserted. In that snippet, NAME is the Value of the Parameter with the Name LogFormatN.Name, and FORMAT is the Value of the Parameter with the Name LogFormatN.Format for the same value of N².

```
# 9: Log Format Snippet
NAME = format {
    Format = 'FORMAT '}
```

Tip: The order in which these Parameters are considered is exactly the numerical ordering implied by N (starting with it being empty). However, each section is generated for all values of N before moving on to the next.

Furthermore, for a given value of N - as before restricted to either the empty string or a natural number on the interval [1,9] -, if a Parameter exists on the cache server’s Profile having this Config File value with the Name LogFilterN.Name, a line of the format NAME = filter.TYPE.('FILTER') will be inserted, where NAME is the Value of the Parameter with the Name LogFilterN.Name, TYPE is the Value of the Parameter with the Name LogFilterN.Type, and FILTER is the Value of the Parameter with the name LogFilterN.Filter³.

Note: When, for a given value of N, a Parameter with the Name LogFilterN.Name exists, but a Parameter with the Name LogFilterN.Type does not exist, the value of TYPE will be accept.

Finally, for a given value of N, if a Parameter exists on the cache server’s Profile having this Config File value with the Name LogObjectN.Filename, the text in Log Object Snippet will be inserted. In that snippet, TYPE is the Value of the Parameter with the Name LogObjectN.Type

² This Value may safely contain double quotes (") as they will be backslash-escaped in the generated output.
³ This Value may safely contain backslashes (\) and single quotes (‘), as they will be backslash-escaped in the generated output.
# 10: Log Object Snippet

```
log.TYPE {
    Format = FORMAT,
    Filename = 'FILENAME',
}
```

**Note:** When, for a given value of N a Parameter with the Name LogObjectN.Filename exists, but a Parameter with the Name LogObjectN.Type does not exist, the value of TYPE in Log Object Snippet will be ascii.

At this point, if the Value of the Parameter with the Name LogObjectN.Type is exactly pipe, a line of the format Filters = FILTERS will be inserted where FILTERS is the Value of the Parameter with the Name LogObjectN.Filters, followed by a line containing only a closing “curly brace” (}) - if and only if said Parameter is not empty. If, however, the Value of the Parameter with the Name LogObjectN.Type is not exactly pipe, then the text in Log Object (not a “pipe”) Snippet is inserted.

# 11: Log Object (not a “pipe”) Snippet

```
RollingEnabled = ROLLING,
RollingIntervalSec = INTERVAL,
RollingOffsetHr = OFFSET,
RollingSizeMb = SIZE
}
```

In this snippet, ROLLING is the Value of the Parameter with the Name LogObjectN. RollingEnabled, INTERVAL is the Value of the Parameter with the Name LogObjectN.RollingIntervalSec, OFFSET is the Value of the Parameter with the Name LogObjectN.RollingOffsetHr, and SIZE is the Value of the Parameter with the Name LogObjectN.SizeMb - all still having the same value of N, and the Config File value logging.config, of course.

**Warning:** The contents of these fields are not validated by Traffic Control - handle with care!

See also:

The Apache Traffic Server documentation for the logging.config configuration file

logging.yaml

This is a YAML-format configuration file used by cache servers that use Apache Traffic Server version 8 or higher - for lower versions, users/administrators/developers should instead be configuring logging.config. This configuration always starts with (after the header) the single line: format:. Afterward, for every Parameter assigned to this Config File with a Name
like LogFormat.N.Name where N is either the empty string or a natural number on the interval [1,9], the YAML fragment shown in Log Format Snippet will be inserted. In this snippet, NAME is the Value of the Parameter with the Name LogFormat.N.Name, and for the same value of N FORMAT is the Value of the Parameter with the Name LogFormat.N.Format.

# 12: Log Format Snippet

```yaml
- name: NAME
  format: 'FORMAT'
```

Tip: The order in which these Parameters are considered is exactly the numerical ordering implied by N (starting with it being empty). However, each section is generated for all values of N before moving on to the next.

After this, a single line containing only filters: is inserted. Then, for each Parameter on the cache server’s Profile with a Name like LogFilter.N.Name where N is either the empty string or a natural number on the interval [1,9], the YAML fragment in Log Filter Snippet will be inserted. In that snippet, NAME is the Value of the Parameter with the Name LogFilter.N.Name, TYPE is the Value of the Parameter with the Name LogFilter.N.Type for the same value of N, and FILTER is the Value of the Parameter with the Name LogFilter.N.Filter for the same value of N.

# 13: Log Filter Snippet

```yaml
- name: NAME
  action: TYPE
  condition: FILTER
```

Note: When, for a given value of N, a Parameter with the Name LogFilter.N.Name exists, but a Parameter with the Name LogFilter.N.Type does not exist, the value of TYPE in Log Filter Snippet will be accept.

At this point, a single line containing only logs: is inserted. Finally, for each Parameter on the cache server’s Profile assigned to this Config File with a Name like LogObject.N.Filename where N is once again either an empty string or a natural number on the interval [1,9] the YAML fragment in Log Object Snippet will be inserted. In this snippet, for a given value of N TYPE is the Value of the Parameter with the Name LogObject.N.Type, FILENAME is the Value of the Parameter with the Name LogObject.N.Filename, FORMAT is the Value of the Parameter with the Name LogObject.N.Format.
# 14: Log Object Snippet

```
- mode: TYPE
  filename: FILENAME
  format: FORMAT
  ROLLING_OR.Filters
```

**Note:** When, for a given value of `N` a Parameter with the `LogObject.N.Filename` exists, but a Parameter with the `LogObject.N.Type` does **not** exist, the value of `TYPE` in *Log Object Snippet* will be ascii.

`ROLLING_OR.Filters` will be one of two YAML fragments based on the Value of the Parameter with the name `LogObject.N.Type`. In particular, if it is exactly `pipe`, then `ROLLING.OR. Filters` will be `filters: [FILTERS]` where `FILTERS` is the Value of the Parameter assigned to this Config File with the `LogObject.N.Filters` for the same value of `N`. If, however, the Value of the Parameter with the `LogObject.N.Type` is **not** exactly `pipe`, `ROLLING.OR.Filters` will have the format given by *Log Object (not a “pipe”) Snippet*. In that snippet, `ROLLING` is the Value of the Parameter with the `LogObject.N.RollingEnabled`, `INTERVAL` is the Value of the Parameter with the `LogObject.N.RollingIntervalSec`, `OFFSET` is the Value of the Parameter with the `LogObject.N.RollingOffsetHr`, and `SIZE` is the Value of the Parameter with the `LogObject.N.RollingSizeMb` - all for the same value of `N` and assigned to the `logging.yaml` Config File, obviously.

# 15: Log Object (not a “pipe”) Snippet

```
rolling_enabled: ROLLING
rolling_interval_sec: INTERVAL
rolling_offset_hr: OFFSET
rolling_size_mb: SIZE
```

See also:

For an explanation of YAML syntax, refer to the official specification thereof. For an explanation of the syntax of a valid Apache Traffic Server `logging.yaml` configuration file, refer to that project’s dedicated documentation.

**logs_xml.config**

This configuration file is somewhat more complex than most Config Files, in that it generates XML document tree segments\(^1\) for each Parameter on the *cache server’s Profile* rather than simply a plain-text line. Specifically, up to ten of the document fragment shown in *LogFormat Snippet* will be inserted, one for each Parameter with this Config File value on the *cache*

\(^1\) The contents of this file are not valid XML, but are rather XML-like so developers writing procedures that will consume and parse it should be aware of this, and note the actual syntax as specified in the Apache Traffic Server documentation for `logs_xml.config`
server’s Profile that has a Name like LogFormat\texttt{N}.Name where \( N \) is either the empty string or a natural number on the range \([1,9]\). In that snippet, the string \texttt{NAME} is actually the Value of the Parameter with the Name LogFormat\texttt{N}.Name. \texttt{FORMAT} is the Value of the Parameter with the Name LogFormat\texttt{N}.Format\textsuperscript{2}, where again \( N \) is either the empty string or a natural number on the interval \([1,9]\) - same-valued \( N \) Parameters are associated.

# 16: LogFormat Snippet

```
<LogFormat>
  <Name = "NAME"/>
  <Format = "FORMAT"/>
</LogFormat>
```

Tip: The order in which these Parameters are considered is exactly the numerical ordering implied by \( N \) (starting with it being empty).

Furthermore, for a given value of \( N \), if a Parameter exists on the cache server’s Profile having this Config File value with the Name LogObject\texttt{N}.Filename, the document fragment shown in \texttt{LogObject Snippet} will be inserted. In that snippet, \texttt{OBJ_FORMAT} is the Value of the Parameter with the Name LogObject\texttt{N}.Format, FILENAME is the Value of the Parameter with the Name LogObject\texttt{N}.Filename, ROLLING is the Value of the Parameter with the Name LogObject\texttt{N}.RollingEnabled, INTERVAL is the Value of the Parameter with the Name LogObject\texttt{N}.RollingIntervalSec, OFFSET is the Value of the Parameter with the Name LogObject\texttt{N}.RollingOffsetHr, SIZE is the Value of the Parameter with the Name LogObject\texttt{N}.RollingSizeMb, and HEADER is the Value of the Parameter with the Name LogObject\texttt{N}.Header - all having the same value of \( N \), and the Config File value logs\_xml\_config, of course.

# 17: LogObject Snippet

```
<LogObject>
  <Format = "OBJ_FORMAT"/>
  <Filename = "FILENAME"/>
  <RollingEnabled = ROLLING/>
  <RollingInterval = INTERVAL/>
  <RollingOffsetHr = OFFSET/>
  <RollingSizeMb = SIZE/>
  <Header = "HEADER"/>
</LogObject>
```

Warning: The contents of these fields are not validated by Traffic Control - handle with care!

See also:
The Apache Traffic Control documentation on the logs\_xml\_config configuration file

Deprecated since version ATCv3.0: This file is only used by Apache Traffic Server version 6.x.
The use of Apache Traffic Server version < 7.1 has been deprecated, and will not be supported in the future. Developers are encouraged to instead configure the `logging.config` configuration file.

**package**

This is a special, reserved Config File that isn’t a file at all. When a Parameter’s Config File is `package`, then its name is interpreted as the name of a package. `ORT` on the server using the `Profile` that has this Parameter will attempt to install a package by that name, interpreting the Parameter’s `Value` as a version string if it is not empty. The package manager used will be `yum(8)`, regardless of system (though the Python version of `ORT` will attempt to use the host system’s package manager - `yum(8)`, `apt(8)` and `pacman` are supported) but that shouldn’t be a problem because only CentOS 7 is supported.

The current implementation of `ORT` will expect Parameters to exist on a cache server’s `Profile` with the `Names astats_over_http and trafficserver` before being run the first time, as both of these are required for a cache server to operate within a Traffic Control CDN. It is possible to install these outside of `ORT` - and indeed even outside of `yum(8)` - but such configuration is not officially supported.

**packages**

This Config File is reserved, and is used by `ORT` to pull bulk information about all of the Parameters with Config File values of `package`. It doesn’t actually correspond to any configuration file.

**parent.config**

This configuration file is generated entirely from `Cache Group` relationships, as well as `Delivery Service` configuration. This file *can* be affected by Parameters on the server’s `Profile` if and only if its `Name` is one of the following:

- `algorithm`
- `qstring`
- `psel.qstring_handling`
- `not_a_parent` - unlike the other Parameters listed (which have a 1:1 correspondence with Apache Traffic Server configuration options), this Parameter affects the generation of `parent` relationships between `cache servers`. When a Parameter with this `Name` and Config File exists on a `Profile` used by a `cache server`, it will not be added as a `parent` of any other `cache server`, regardless of `Cache Group` hierarchy. Under ordinary circumstances, there’s no real reason for this Parameter to exist.

Additionally, `Delivery Service Profiles` can have special Parameters with the Name “`mso.parent_retry`” to `Configure Multi-Site Origin`.

**See also:**

**2.1. Traffic Control Overview**
To see how the *Values* of these Parameters are interpreted, refer to the Apache Traffic Server documentation on the parent.config configuration file.

**plugin.config**

For each Parameter with this Config File value on the same Profile, a line in the resulting configuration file is produced in the format `NAME VALUE` where `NAME` is the Parameter’s Name with trailing characters matching the regular expression `__\d+` stripped out and `VALUE` is the Parameter’s Value.

**Caution:** In order for Parameters for Config Files relating to Apache Traffic Server plugins - e.g. `regex_revalidate.config` - to have any effect, a Parameter must exist with this Config File value to instruct Apache Traffic Server to load the plugin. Typically, this is more easily achieved by assigning these Parameters to *The GLOBAL Profile* than on a server-by-server basis.

See also:

The Apache Traffic server documentation on the plugin.config configuration file explains what Value and Name a Parameter should have to be valid.

**rascal.properties**

This Config File is meant to be on Parameters assigned to either Traffic Monitor Profiles or cache server Profiles. Its allowed Parameter Names are all configuration options for Traffic Monitor. The Names with meaning are as follows.

**health.threshold.loadavg** The Value of this Parameter sets the “load average” above which the associated Profile’s cache server will be considered “unhealthy”.

See also:

*Health Protocol*

See also:

The definition of a “load average” can be found in the documentation for the Linux/Unix command `uptime(1)`.

**Caution:** If more than one Parameter with this Name and Config File exist on the same Profile with different Values, the actual Value used by any given Traffic Monitor instance is undefined (though it will be the Value of one of those Parameters).

**health.threshold.availableBandwidthInKbps** The Value of this Parameter sets the amount of bandwidth (in kilobits per second) that Traffic Control will try to keep available on the cache server. For example a Value of “>1500000” indicates that the cache server will be
marked “unhealthy” if its available remaining bandwidth on the network interface used by the caching proxy falls below 1.5Gbps.

See also:

*Health Protocol*

**Caution:** If more than one Parameter with this *Name* and Config File exist on the same *Profile* with different *Values*, the actual *Value* used by any given Traffic Monitor instance is undefined (though it will be the *Value* of one of those Parameters).

**records.config**

For each Parameter with this Config File value on the same *Profile*, a line in the resulting configuration file is produced in the format *NAME VALUE* where *NAME* is the Parameter’s *Name* with trailing characters matching the regular expression __\d+$/ stripped out and *VALUE* is the Parameter’s *Value*.

See also:

The Apache Traffic Server records.config documentation

**regex_remapanything.config**

Config Files matching this pattern - where *anything* is zero or more characters - are generated entirely from *Delivery Service* configuration, which cannot be affected by any Parameters (except “location”).

See also:

For the syntax of configuration files for the “Regex Remap” plugin, see the Regex Remap plugin’s official documentation. For instructions on how to enable a plugin, consult, the plugin.config documentation.

**regex_revalidate.config**

This configuration file can only be affected by the special maxRevalDurationDays, which is discussed in the *The GLOBAL Profile* section.

See also:

For the syntax of configuration files for the “Regex Revalidate” plugin, see the Regex Revalidate plugin’s official documentation. For instructions on how to enable a plugin, consult, the plugin.config documentation.
remap.config

This configuration file can only be affected by one Parameter on a Profile assigned to a Delivery Service. Then, for every Parameter assigned to that Profile that has the Config File value “cachekey.config” - NOT this Config File -, a parameter will be added to the line for that Delivery Service like so: pparam=--Name=Value where Name is the Parameter’s Name, and Value is its Value.

See also:
For an explanation of the syntax of this configuration file, refer to the Apache Traffic Server remap.config documentation.

set_dscpanything.config

Configuration files matching this pattern - where anything is a string of zero or more characters is generated entirely from a “location” Parameter.

Tip: anything in that Config File name only has meaning if it is a natural number - specifically, one of each value of DSCP on every Delivery Service to which the cache server using the Profile on which the Parameter(s) exist(s).

ssl_multicert.config

This configuration file is generated from the SSL keys of Delivery Services, and is unaffected by any Parameters (except “location”)

See also:
The official ssl_multicert.config documentation

storage.config

This configuration file can only be affected by a handful of Parameters. If a Parameter with the Name “Drive Prefix” exists the generated configuration file will have a line inserted in the format PREFIXLETTER volume=1 for each letter in the comma-delimited list that is the Value of the Parameter on the same Profile with the Name “Drive Letters”, where PREFIX is the Value of the Parameter with the Name “Drive Prefix”, and LETTER is each of the aforementioned letters in turn. Additionally, if a Parameter on the same Profile exists with the Name “RAM Drive Prefix” then for each letter in the comma-delimited list that is the Value of the Parameter on the same Profile with the Name “RAM Drive Letters”, a line will be generated in the format PREFIXLETTER volume=i where PREFIX is the Value of the Parameter with the Name “RAM Drive Prefix”, LETTER is each of the aforementioned letters in turn, and i is 1 if and only if a Parameter does not exist on the same Profile with the Name “Drive Prefix” and is 2 otherwise. Finally, if a Parameter exists on the same Profile with the Name
“SSD Drive Prefix”, then a line is inserted for each letter in the comma-delimited list that is the \textit{Value} of the Parameter on the same \textit{Profile} with the \textit{Name} “SSD Drive Letters” in the format \texttt{PREFIXLETTER volume=i} where \texttt{PREFIX} is the \textit{Value} of the Parameter with the \textit{Name} “SSD Drive Prefix”, \texttt{LETTER} is each of the aforementioned letters in turn, and \texttt{i} is 1 \textit{if and only if both} a Parameter with the \textit{Name} “Drive Prefix” and a Parameter with the \textit{Name} “RAM Drive Prefix” \textit{don’t exist on the same Profile}, or 2 if only \texttt{one} of them exists, or otherwise 3.

\textbf{See also:}

The Apache Traffic Server storage.config file documentation.

\texttt{traffic_stats.config}

This Config File value is only handled specially when the \textit{Profile} to which it is assigned is of the special TRAFFIC_STATS \textit{Type}. In that case, the \textit{Name} of any Parameters with this Config File is restrained to one of “CacheStats” or “DsStats”. When it is “Cache Stats”, the \textit{Value} is interpreted specially based on whether or not it starts with “ats.”. If it does, then what follows must be the name of one of the \textbf{core} Apache Traffic Server statistics. This signifies to Traffic Stats that it should store that statistic for \textit{cache servers} within Traffic Control. Additionally, the special statistics “bandwidth”, “maxKbps” are supported as \textit{Names} - and in fact it is suggested that they exist in every Traffic Control deployment.

When the Parameter \textit{Name} is “DSStats”, the allowed \textit{Values} are:

- kbps
- status\_4xx
- status\_5xx
- tps\_2xx
- tps\_3xx
- tps\_4xx
- tps\_5xx
- tps\_total

\textbf{See also:}

For more information on the statistics gathered by Traffic Stats, see \textit{Traffic Stats Administration}. For information about how these statics are gathered, consult the only known documentation of the “astats\_over\_http” Apache Traffic Server plugin: \texttt{traffic_server/plugins/astats\_over\_http/README.md}.

\texttt{sysctl.config}

For each Parameter with this Config File value on the same \textit{Profile}, a line in the resulting configuration file is produced in the format \texttt{NAME = VALUE} where \texttt{NAME} is the Parameter’s \textit{Name} with trailing characters matching the regular expression \texttt{\_\_\d+\$} stripped out and \texttt{VALUE} is the Parameter’s \textit{Value}. 

\section{2.1. Traffic Control Overview}
uri_signing Anything.config

Config Files matching this pattern - where anything is zero or more characters - are generated entirely from the URI Signing Keys configured on a Delivery Service through either the Traffic Ops API or the Delivery Services view in Traffic Portal.

See also:

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.

url_sig Anything.config

Config Files that match this pattern - where anything is zero or more characters - are mostly generated using the URL Signature Keys as configured either through the Traffic Ops API or the Delivery Services view in Traffic Portal. However, if no such keys have been configured, they may be provided by fall-back Parameters. In this case, for each Parameter on assigned to this Config File on the same Profile a line is inserted into the resulting configuration file in the format NAME = VALUE where NAME is the Parameter’s Name and VALUE is the Parameter’s Value.

See also:

The Apache Trafficserver documentation for the url_sig plugin.

volume.config

This Config File is peculiar in that it depends only on the existence of Parameters, and not each Parameter’s actual Value. The Parameters that affect the generated configuration file are the Parameters with the Names “Drive Prefix”, “RAM Drive Prefix”, and “SSD Drive Prefix”. Each of these Parameters must be assigned to the storage.config Config File - NOT this Config File - and, of course, be on the same Profile. The contents of the generated Config File will be between zero and three lines (excluding headers) where the number of lines is equal to the number of the aforementioned Parameters that actually exist on the same Profile. Each line has the format volume=i scheme=http size=SIZE% where i is a natural number that ranges from 1 to the number of those Parameters that exist. SIZE is 100/N - where N is the number of those special Parameters that exist - truncated to the nearest natural number, e.g. 100/3 = 33.

See also:

The Apache Traffic Server volume.config file documentation.

ID

An integral, unique identifier for a Parameter. Note that Parameters must have a unique combination of Config File, Name, and Value, and so those should be used for identifying a unique Parameter whenever possible.
Implementation Detail

If two Profiles have been assigned Parameters that have the same values for Config File, Name, and Value then Traffic Ops actually only stores one Parameter object and merely links it to both Profiles. This can be seen by inspecting the Parameters’ IDs, as they will be the same. There are many cases where a user or developer must rely on this implementation detail, but both are encouraged to do so only when absolutely necessary.

Name

The Name of a Parameter has different meanings depending on the type of any and all Profiles to which it is assigned, as well as the Config File to which the Parameter belongs, but most generally it is used in Apache Traffic Server configuration files as the name of a configuration option in a name/value pair. Traffic Ops interprets the Name and Value of a Parameter in intelligent ways depending on the type of object to which the Profile using the Parameter is assigned. For example, if Config File is records.config and the Parameter’s Profile is assigned to a cache server, then a single line is placed in the configuration file specified by Config File, and that line will have the contents Name Value. However, if the Config File of the Parameter is something without special meaning to Traffic Ops e.g. “foo”, then a line containing only the Parameter’s Value would be inserted into that file (presuming it also has a Parameter with a Name of “location” and a Config File of “foo”). Additionally, there are a few Names that are treated specially by Traffic Control.

location The Value of this Parameter is to be interpreted as a path under which the configuration file specified by Config File shall be found (or written, if not found). Any configuration file that is to exist on a server must have an associated “location” Parameter, even if the contents of the file cannot be affected by Parameters.

header If the Profile containing this Parameter is assigned to a server, and if the Config File is not one of the special values that Traffic Ops uses to determine special syntax formatting, then the Value of this Parameter will be used instead of the typical Traffic Ops header - unless it is the special string “none”, in which case no header will be inserted at all.

Caution: If a single Profile has multiple “location” Parameters for the same Config File with different Values, the actual location of the generated configuration file is undefined (but will be one of those Parameters’ Values).

Caution: If a single Profile has multiple “header” Parameters for the same Config File with different Values, the actual header is undefined (but will be one of those Parameters’ Values).
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’.

Value

In general, a Parameter’s Value can be anything, and in the vast majority of cases the Value is in no way validated by Traffic Control. Usually, though, the Value has a special meaning depending on the values of the Parameter’s Config File and/or Name.
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

Traffic Ops is quite possible the single most complex and most important Traffic Control component. It has many different configuration options that affect a wide range of other components and their interactions.

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 `yum(8)` repositories
- Access to The Comprehensive Perl Archive Network (CPAN)

Guide

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

   See also:

   For more information on installing PostgreSQL, see their documentation.

   # 18: 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'
```

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 192.0.2.1 add the line `host all all 192.0.2.1/32 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
# 19: Starting PostgreSQL with `systemd(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 - or download a pre-built release from the Apache Continuous Integration server.

5. Install a PostgreSQL client on the Traffic Ops host

   # 20: 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.

   # 21: 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

   # 22: 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.
# 23: Creating the Traffic Ops User and Database

```
$ psql -U postgres -h pg -c "CREATE USER traffic_ops WITH ENCRYPTED PASSWORD 'tcr0cks';"
Password for user postgres:
CREATE ROLE
$ createdb traffic_ops --owner traffic_ops -U postgres -h pg
Password:
```

9. Now, run the following command as the root user (or with `sudo(8)`): `/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.

# 24: 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]: pg
Database server hostname IP or FQDN: pg
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 ==========
Database server root (admin) user [postgres]:
Database server root (admin) user: postgres
Password for database server admin:
```

(continues on next page)
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:
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:
=========/opt/traffic_ops/install/data/json/profiles.json=========
Traffic Ops url [https://localhost]:
Traffic Ops url: https://localhost
Human-readable CDN Name. (No whitespace, please):
[kabletown_cdn]: 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
download/geolite/database/GeoLite2-City.mmdb.gz
Resolving geolite.maxmind.com (geolite.maxmind.com)...
2400:cb00:2048:1::6810:262f, ...
Connecting to geolite.maxmind.com (geolite.maxmind.com)...
2400:cb00:2048:1::6810:262f, 104.16.38.47, ...
(continues on next page)
Starting Traffic Ops
Restarting traffic_ops (via systemctl):
  → [ OK ]
Waiting for Traffic Ops to restart
Success! Postinstall complete.

Table 41: 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 or FQDN</td>
<td>The hostname of the database server (pg in the example).</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) user name</td>
<td>Privileged database user that has permission to create the database and user for Traffic Ops...</td>
</tr>
<tr>
<td>Database server root (admin) user password</td>
<td>The password for the privileged database user.</td>
</tr>
<tr>
<td>Traffic Ops URL</td>
<td>The URL to connect to this instance of Traffic Ops, usually <a href="https://Traffic">https://Traffic</a> Ops host</td>
</tr>
<tr>
<td>Human-readable CDN Name</td>
<td>The name of the first CDN which Traffic Ops will manage.</td>
</tr>
<tr>
<td>DNS sub-domain for which your CDN is authoritative</td>
<td>The DNS domain that will be delegated to this Traffic Control CDN.</td>
</tr>
<tr>
<td>Administration username for Traffic Ops</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>Password for the admin user</td>
<td>The password for the administrative Traffic Ops user.</td>
</tr>
</tbody>
</table>
Upgrading

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.

**Tip:** In order to upgrade to the latest version of Traffic Ops, please be sure that you have first upgraded to the latest available minor or patch version of your current release. For example, if your current Traffic Ops version is 3.0.0 and version 3.1.0 is available, you must first upgrade to 3.1.0 before proceeding to upgrade to 4.0.0. (Specifically, this means running all migrations, `traffic_ops/app/db/seeds.sql`, and `traffic_ops/app/db/patches.sql` for the latest of your current major version - which should be handled by the `admin` tool). The latest migration available before the release of 4.0.0 (pending at the time of this writing) was `traffic_ops/app/db/migrations/20180814000625_remove_capabilities_for_reseed.sql`, so be sure that migrations up to this point have been run before attempting to upgrade Traffic Ops.

**See also:**

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

# 25: 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 *Guide* for instructions on running the `postinstall` script. Once the `postinstall` script, has finished, run the following command as the root user (or with `sudo(8)`): `systemctl start traffic_ops` to start the service.

Running

Currently, Traffic Ops consists of two programs, as it is in the middle of a transition from one code-base to another. It is not recommended that either one be run on its own. Also, while this section contains instructions for running each piece manually, the only truly supported method is via `systemd(8)`, e.g. `systemctl start traffic_ops` (this method also starts both programs properly and uses their default configuration file locations).

**Legacy Perl Script**

The old code base was written in Perl, and is executed as a Perl script. When *Installing* Traffic Ops, the `Hypnotoad` program should be installed at `/opt/traffic_ops/app/local/bin/hypnotoad`. To run Traffic Ops:
1. Change directories to `/opt/traffic_ops/app` - it may not seem like it, but this step is important. *Do not skip it.*

2. Set the `PERL5LIB` environment variable to exactly `/opt/traffic_ops_extensions/private/lib:/opt/traffic_ops/app/lib:/opt/traffic_ops/app/lib/perl5:$PERL5LIB`. That last portion is especially important, to preserve the paths to system libraries - it should expand to whatever `PERL5LIB` was prior to this change.

3. Run the command `/opt/traffic_ops/app/local/bin/hypnotoad script/cdn`

The script takes no options other than the ones accepted by Hypnotoad itself, and it just expects to get its configuration from the standard file locations (covered in *Configuring*).

**MOJO_MODE**

This sets the “mode” of the Traffic Ops Mojolicious application. Effectively, this chooses the set of configuration files it will consult. The default value is “development”, and the possible values are:

- development
- integration
- production
- test

**traffic_ops_golang**


`--cfg CONFIG_PATH`

This *mandatory* command line flag specifies the absolute or relative path to the configuration file to be used by Traffic Ops - `cdn.conf`.

`--dbcfg DB_CONFIG_PATH`

This *mandatory* command line flag specifies the absolute or relative path to a configuration file used by Traffic Ops to establish connections to the PostgreSQL database - `database.conf`

`--plugins`

List the installed plugins and exit.

**Note:** This only accounts for the plugins for the Go version, extensions to the *Legacy Perl Script* are not accounted for.

`--api-routes`

Print information about all API routes and exit. If also used with the `--cfg` option, also print out the configured routing blacklist information from `cdn.conf`.  

84 Chapter 3. Administrator’s Guide
Apache Traffic Control Documentation, Release 4.0.0

Note: This only accounts for routes in the Go version, API routes in Perl but not in Go are not included.

--riakcfg TRAFFIC_VAULT_CONFIG_PATH
This mandatory command line flag specifies the absolute or relative path to a configuration file used by Traffic Ops to establish connections to Traffic Vault - *riak.conf*

Implementation Detail
The name of this flag is derived from the current database used in the implementation of Traffic Vault - Riak KV.

--version
Print version information and exit.

Configuring
The main *traffic_ops_golang* binary and the *Legacy Perl Script* use the same set of configuration files (mostly).

Configuration Files

cdn.conf

This file deals with the configuration parameters of running Traffic Ops itself. It is a JSON-format set of options and their respective values. For the *Legacy Perl Script* to work with this file, it must be in its default location at `/opt/traffic_ops/app/conf/cdn.conf`, but *traffic_ops_golang* will use whatever file is specified by its `--cfg` option. The keys of the file are described below.

- **geniso** This object contains configuration options for system ISO generation.
  - **iso_root_path** Sets the filesystem path to the root of the ISO generation directory. For default installations, this should usually be set to `/opt/traffic_ops/app/public`.

See also:
*Generate ISO*

- **hypnotoad** This is a group of options used nearly exclusively by the *Legacy Perl Script* (as the name implies).

See also:
For a complete description of this section of the configuration, consult *The Mojolicious Cookbook*’s section on Hypnotoad.
**group** Sets the UNIX group as which the *Legacy Perl Script* will run. By default this should be the `trafficops` group which is created when the Traffic Ops RPM is installed.

**heartbeat_timeout** From the Hypnotoad documentation: “The maximum amount of time in seconds before a worker without a heartbeat will be stopped gracefully”.

**listen** This must always be an array containing a single string. This very odd string is apparently a URL. The scheme of the URL specifies the protocol Traffic Ops will use to listen for incoming connections (`traffic_ops_golang` ignores that and always uses HTTPS). The host of the URL tells Traffic Ops on what hostname to listen (`traffic_ops_golang` understands this as the hostname of the *Legacy Perl Script* server), and likewise the port sets the port on which it listens (the port is only used for the *Legacy Perl Script* - `traffic_ops_golang` uses the setting of `traffic_ops_golang.port`). The “cert” query parameter sets the location of the SSL certificate to use for encrypting connections, while the “key” query parameter is the certificate’s corresponding private key. The default configuration file also has the “verify” query parameter which sets the TLS verification mode - 0 for no verification of certificates, 1 to verify them (`traffic_ops_golang` ignores this and uses `traffic_ops_golang.insecure` instead). Finally, a list of TLS ciphers to offer to incoming clients of the *Legacy Perl Script* delimited by commas can be given in the “ciphers” query parameter.

**pid_file** When the *Legacy Perl Script* starts as a daemon, it will write its PID (Process ID) to this file - so be sure that user/group can properly access it! The default setting is `/var/run/traffic_ops.pid`.

**user** Sets the UNIX user as whom the *Legacy Perl Script* will run. By default this should be the `trafficops` user who is created when the Traffic Ops RPM is installed.

**workers** This is used only by the *Legacy Perl Script*, and sets the number of concurrent HTTP server workers allowed.

**inactivity_timeout** This was used by the *Legacy Perl Script* to set timeouts on idle client connections to Traffic Ops - the exact operation (and even units) of this configuration option is unknown. `traffic_ops_golang` ignores this field.

**influx_db_conf_path** An optional field which gives `traffic_ops_golang` the absolute or relative path to an `influxdb.conf` file. Default if not specified is a file named `influxdb.conf` in the same directory as this cdn.conf file.

New in version 4.0.
Warning: While relative paths are allowed, they are discouraged, as the path will be relative to the working directory of the `traffic_ops_golang` process itself, not relative to the `cdn.conf` configuration file, which can be confusing.

**ldap_conf_location** An optional field which gives `traffic_ops_golang` the absolute or relative path to an `ldap.conf` file. Default if not specified is a file named `ldap.conf` in the same directory as this `cdn.conf` file.

Warning: While relative paths are allowed, they are discouraged, as the path will be relative to the working directory of the `traffic_ops_golang` process itself, not relative to the `cdn.conf` configuration file, which can be confusing.

**lets_encrypt** New in version 4.1.

- **user_email** An optional email address to create an account with Let’s Encrypt or to receive expiration updates.
- **send_expiration_email** A boolean option to send email summarizing certificate expiration status.
- **convert_self_signed** A boolean option to convert self signed to Let’s Encrypt certificates as they expire. This only works for certificates labeled as Self Signed in the Certificate Source field.
- **renew_days_before_expiration** Set the number of days before expiration date to renew certificates.
- **environment** This specifies which Let’s Encrypt environment to use: ‘staging’ or ‘production’. It defaults to ‘production’.

**portal** This section provides information regarding a connected UI with which users interact, so that emails can include links to it.

- **base_url** This URL should be the root and/or landing page of the UI. For Traffic Portal instances, this should include the fragment part of the URL, e.g. `https://trafficportal.infra.ciab.test/#!/`.
- **docs_url** The actual use of this URL is unknown, but supposedly it ought to point to the documentation for the Traffic Control instance. It’s hard to imagine a fantastic reason this shouldn’t just always be `https://traffic-control-cdn.readthedocs.io`
- **email_from** Most emails sent from the Traffic Ops server will use `to.email_from`, but specifically password reset requests (which contain a link to a fragment under `portal.base_url`) will instead use this as the value of their `From` field.
pass_reset_path A path to be added to base_url that is the URL of the UI’s password reset interface. For Traffic Portal instances, this should always be set to “user”.

user_register_path A path to be added to base_url that is the URL of the UI’s new user registration interface. For Traffic Portal instances, this should always be set to “user”.

riak_conf_path An optional absolute or relative path to riak.conf. If this field is not defined, is null, or is an empty string (""), Traffic Ops will not be able to connect to Traffic Vault.

Caution: If Traffic Ops is unable to connect to Traffic Vault, many of its core features will not function. In particular, it will be impossible to create Delivery Services that use HTTPS or DNSSEC.

Implementation Detail

The name of this field is derived from the current database used in the implementation of Traffic Vault - Riak KV.

secrets This is an array of strings, which cannot be empty. The first secret in the array is used to encrypt Traffic Ops authentication cookies - multiple Traffic Ops instances serving the same CDN need to share secrets in order for users logged into one to be able to use their cookie as authentication with other instances.

smtp This optional section contains options for connecting to and authenticating with an SMTP (Simple Mail Transfer Protocol) server for sending emails. If this section is undefined (or if enabled is explicitly false), Traffic Ops will not be able to send emails and certain Traffic Ops API endpoints that depend on that functionality will fail to operate.

New in version 4.0.

address This is the address of the SMTP which will be used to send emails. Should include the port number, e.g. "localhost:25" for sendmail(8) on the Traffic Ops server.

enabled A boolean flag that determines whether or not connection to an SMTP ought to be allowed. Whatever the settings of the other fields in the smtp object, email cannot and will not be sent if this is false.

password The password to be used when authenticating with the SMTP server.

user The name of the user to be used when authenticating with the SMTP server.
Note: The SMTP integration currently only supports Login Auth.

to Contains information to identify Traffic Ops in a network sense.

base_url This field is used to identify the location for the now-removed Traffic Ops UI. It no longer serves any purpose.

e-mail_from Sets the address that will appear in the From field of Emails sent by Traffic Ops.

no_account_found_msg When a password reset is requested for an email address not registered to any known user, this is the message that will be sent to that email address.

traffic_ops_golang This group configuration options is used exclusively by traffic_ops_golang.

backend_max_connections This optional object, if declared, is a map of back-end service names to the maximum number of allowed concurrent connections to them from the Traffic Ops server. Currently, the only used key is "mojolicious", which sets the maximum allowed connections to the server running the Legacy Perl Script. If that key is missing - or if this entire optional object is missing - it will default to the value of MojoliciousConcurrentConnectionsDefault.

crconfig_emulate_old_path An optional boolean that controls the value of a part of Snapshots that report what Traffic Ops API endpoint is used to generate Snapshots. If this is true, it forces Traffic Ops to report that a legacy, deprecated endpoint is used, whereas if it's false Traffic Ops will report the actual, current endpoint. Default if not specified is false.

Deprecated since version 3.0: Future versions of Traffic Ops will not support this legacy configuration option, and will always report the current endpoint.

crconfig_snapshot_use_client_request_host An optional boolean which controls the value of the Traffic Ops server's URL as inserted into Snapshots. If this is true, then the value used will be taken from the Host header of the request that generated the Snapshot. If it's false, then it will instead use the value of the global “tm.url” Parameter. Default if not specified is false.

Deprecated since version 3.0: Future versions of Traffic Ops will not support this legacy configuration option, and will always use the global “tm.url” Parameter.

db_conn_max_lifetime_seconds An optional field that sets the maximum lifetime in seconds of any given connection to the Traffic Ops Database. If set to zero, connections are held open
until explicitly closed. Default if not specified is the value of DBConnMaxLifetimeSecondsDefault.

**db_max_idle_connections** An optional limit on the number of connections to the Traffic Ops Database to keep alive while idle. If this is less than max_db_connections, that number will be used instead - *even if this field is unset and using its default*. Default if not specified is the value of DBMaxIdleConnectionsDefault.

**db_query_timeout_seconds** An optional field specifying a timeout on database transactions (not actually single queries in most cases) within API route handlers. Effectively this is a timeout on a single handler’s ability to interact with the Traffic Ops Database. Default if not specified is the value of DefaultDBQueryTimeoutSecs.

**idle_timeout** An optional timeout in seconds for idle client connections to Traffic Ops. If set to zero, the value of read_timeout will be used instead. If both are zero, then the value of read_header_timeout will be used. If all three fields are zero, there is no timeout and connections will be kept alive indefinitely - *not* recommended. Default if not specified is zero.

**insecure** An optional boolean which, if set to true will cause Traffic Ops to skip verification of client certificates whenever necessary/possible. If set to false, the normal verification behavior is exhibited. Default if not specified is false.

**log_location_debug** This optional field, if specified, should either be the location of a file to which debug-level output will be logged, or one of the special strings "stdout" which indicates that STDOUT should be used, "stderr" which indicates that STDERR should be used or "null" which indicates that no output of this level should be generated. An empty string ("") and literally null are equivalent to "null". Default if not specified is "null".

**log_location_error** This optional field, if specified, should either be the location of a file to which error-level output will be logged, or one of the special strings "stdout" which indicates that STDOUT should be used, "stderr" which indicates that STDERR should be used or "null" which indicates that no output of this level should be generated. An empty string ("") and literally null are equivalent to "null". Default if not specified is "null". This field is also used to determine where server profiling statistics are written. Assuming profiling_enabled is true and profiling_location is unset, if this field’s value is given as a path to a regular file, a file named profiling will be written to the same directory containing the profiling information -
overwriting any existing files by that name.

**log_location_event** This optional field, if specified, should either be the location of a file to which event-level output will be logged, or one of the special strings "stdout" which indicates that STDOUT should be used, "stderr" which indicates that STDERR should be used or "null" which indicates that no output of this level should be generated. An empty string (""") and literally null are equivalent to "null". Default if not specified is "null".

**log_location_info** This optional field, if specified, should either be the location of a file to which informational-level output will be logged, or one of the special strings "stdout" which indicates that STDOUT should be used, "stderr" which indicates that STDERR should be used or "null" which indicates that no output of this level should be generated. An empty string (""") and literally null are equivalent to "null". Default if not specified is "null".

**log_location_warning** This optional field, if specified, should either be the location of a file to which warning-level output will be logged, or one of the special strings "stdout" which indicates that STDOUT should be used, "stderr" which indicates that STDERR should be used or "null" which indicates that no output of this level should be generated. An empty string (""") and literally null are equivalent to "null". Default if not specified is "null".

**max_db_connections** An optional limit on the number of allowed concurrent connections to the Traffic Ops Database. If it is less than or equal to zero, there is no limit. Default if not specified is zero.

**oauth_client_secret** An optional secret string to be shared with OAuth-capable clients attempting to authenticate via OAuth. The default behavior if this is not defined - or is an empty string ("") or null is to disallow authentication via OAuth.

**Warning:** OAuth support in Traffic Ops is still in its infancy, so most users are advised to avoid defining this field without good cause.

**plugins** An optional array of enabled plugin names. These names must be unique. Note that a plugin that is installed will not be used unless its name appears in this list - thus “enabling” it. If not specified no plugins will be enabled.

**plugin_config** This optional object maps plugin names - which must appear in the plugins array - to arbitrary JSON con-
figurations for said plugins. It is up to the plugins themselves to parse these configurations. The default if not specified is no configuration information, somewhat obviously.

**plugin_shared_config** This optional object is just an arbitrary JSON object that is converted into a native object and made available to any and all loaded and enabled plugins. A typical use-case for this field is avoiding repetition of identical configuration in `plugin_config`. The default if not specified is null.

**port** Sets the port on which Traffic Ops will listen for incoming connections.

**profiling_enabled** An optional boolean which, if `true` will enable the gathering of profiling statistics on the Traffic Ops server. Default if not specified is `false`.

**profiling_location** An optional string which, if set, should be the absolute path (relative paths are allowed but not recommended) to a file where profiling statistics for the Traffic Ops server will be written. If `profiling_enabled` is `true` but this is not specified, or is an empty string ("") or `null`, then a file named “profiling” will be created or overwritten in the same directory as the file specified in `log_location_error`. If that file is not a regular file, then Traffic ops will instead create a temporary directory and write profiling statistics to a file named “profiling” within that directory.

**proxy_keep_alive** An optional field which specifies the keep-alive period in seconds for connections from Traffic Ops back to the `Legacy Perl Script`. If set to a negative number, keep-alives will be disabled. Default if not specified is zero.

**proxy_read_handler_timeout** An optional timeout in seconds for Traffic Ops to wait for a request after writing a request to the `Legacy Perl Script`. If set to zero, Traffic Ops will wait until it gets a response (i.e. no timeout - not recommended). Default if not specified is zero.

**proxy_timeout** An optional timeout in seconds for connections from Traffic Ops back to the `Legacy Perl Script`. If set to zero, there is no timeout. Default if not specified is zero.

**proxy_tls_timeout** An optional field that sets the timeout in seconds for TLS handshakes from Traffic Ops to the `Legacy Perl Script`. If set to zero, there is no timeout. The default if not specified is zero.

**read_header_timeout** An optional timeout in seconds before which Traffic Ops must be able to finish reading the headers of an incoming request or it will drop the connection. If set to
zero, there is no timeout. Default if not specified is zero.

**read_timeout** An optional timeout in seconds before which Traffic Ops must be able to finish reading an entire incoming request (including body) or it will drop the connection. If set to zero, there is no timeout. Default if not specified is zero.

**riak_port** An optional field that sets the port on which Traffic Ops will try to contact Traffic Vault for storage and retrieval of sensitive encryption keys.

### Implementation Detail

The name of this field is derived from the current database used in the implementation of Traffic Vault - *Riak KV*.

**whitelisted_oauth_url** An optional array of URLs which are allowed to authenticate Traffic Ops users via OAuth. The default behavior if this field is not defined is to not allow OAuth authentication.

**Warning:** OAuth support in Traffic Ops is still in its infancy, so most users are advised to avoid defining this field without good cause.

**write_timeout** An optional timeout in seconds set on handlers. After reading a request’s header, the server will have this long to send back a response. If set to zero, there is no timeout. Default if not specified is zero.

**routing_blacklist** Optional configuration for explicitly routing requests to TO-Perl via *perl_routes* (only routes that are hardcoded to be able to bypass to TO-Perl – not all Go routes can be bypassed to Perl) or explicitly disabling any routes via *disabled_routes*.

New in version 4.0.

**perl_routes** A list of API route IDs to be handled by TO-Perl (rather than by the matching routes in *traffic_ops_golang*). This list can only contain IDs for routes that are on the hardcoded (within *traffic_ops_golang*) whitelist of routes that can be bypassed to TO-Perl. This configuration is meant to allow falling back to TO-Perl for routes that have been rewritten to TO-Go but have been found to contain regressions. In order to find which routes can be bypassed to TO-Perl, run `. /traffic_ops_golang` using the
---api-routes option. This will print out information about all API routes in traffic_ops_golang, including route IDs, paths, and whether or not routes can be bypassed to Perl. In general, the whitelist will contain only routes that have recently been rewritten to Go but not yet included in a release, and only if the Go route has not deviated from its corresponding Perl route in a way that would make it dangerous to fall back to. This whitelist should be expected to change as Go routes become “vetted” in a release. Once TO-Perl is removed, this field will be removed/ignored.

**disabled_routes** A list of API route IDs to disable. Requests matching these routes will receive a 503 response. To find the route ID for a given path you would like to disable, run ./traffic_ops_golang using the --api-routes option to view all the route information, including route IDs and paths.

**ignore_unknown_routes** If false (default) return an error and prevent startup if unknown route IDs are found. Otherwise, log a warning and continue startup.

### Example cdn.conf

```json
{
    "hypnotoad" : {
        "listen" : [
            "https://[::]:60443?cert=/etc/pki/tls/certs/localhost.crt&key=/etc/pki/tls/private/localhost.key&verify=0x00&ciphers=AES128-GCM-SHA256:HIGH:!RC4:!MD5:!aNULL:!EDH:!ED"
        ],
        "user" : "trafops",
        "group" : "trafops",
        "heartbeat_timeout" : 20,
        "pid_file" : "/var/run/traffic_ops.pid",
        "workers" : 12
    },
    "traffic_ops_golang" : {
        "port" : "443",
        "proxy_timeout" : 60,
        "proxy_keep_alive" : 60,
        "proxy_tls_timeout" : 60,
        "proxy_read_header_timeout" : 60,
        "read_timeout" : 60,
        "read_header_timeout" : 60,
        "write_timeout" : 60,
        "idle_timeout" : 60,
        "log_location_error": "/var/log/traffic_ops/error.log",
        "log_location_error": "/var/log/traffic_ops/error.log",
    }
}
```

(continues on next page)
"log_location_warning": "",
"log_location_info": "",
"log_location_debug": "",
"log_location_event": "/var/log/traffic_ops/access.log",
"max_db_connections": 20,
"db_max_idle_connections": 15,
"db_conn_max_lifetime_seconds": 60,
"db_query_timeout_seconds": 20,
"backend_max_connections": {
    "mojolicious": 4
},
"whitelisted_oauth_urls": [],
"oauth_client_secret": "",
"routing_blacklist": {
    "ignore_unknown_routes": false,
    "perl_routes": [],
    "disabled_routes": []
},
"profiling_enabled": false

"cors": {
    "access_control_allow_origin": "*"
},
"to": {
    "base_url": "http://localhost:3000",
    "email_from": "no-reply@traffic-ops-domain.com",
    "no_account_found_msg": "A Traffic Ops user account is required for access. Please contact your Traffic Ops administrator."
},
"portal": {
    "base_url": "https://localhost/#!/",
    "docs_url": "https://traffic-control-cdn.readthedocs.io/en/latest/admin/traffic_portal/usingtrafficportal.html",
    "email_from": "no-reply@traffic-portal-domain.com",
    "pass_reset_path": "user",
    "user_register_path": "user"
},
"secrets": [
    "mONKEYDOmONKEYSEE."
],
"geniso": {
    "iso_root_path": "/opt/traffic_ops/app/public"
},
"inactivity_timeout": 60,
"smtp": {
    "enabled": false,
    "user": "",
    "password": "",
    "from": "no-reply@traffic-ops-domain.com",
    "host": "smtp.example.com",
    "port": 587,
    "tls": true
}
(continues on next page)
database.conf

This file deals with configuration of the Traffic Ops Database; in particular it tells Traffic Ops how to connect with the database for its current environment. traffic_ops_golang will read this file in from the path pointed to by its `--dbcfg` flag, but the Legacy Perl Script expects it to be at `/directory where cdn.conf is located/value of $MOJO_MODE/database.conf` where the value of `$MOJO_MODE` part is literally the value of the `MOJO_MODE` environment variable. database.conf is encoded as a JSON object, and its keys are described below.

**dbname** The name of the PostgreSQL database used. Typically different databases are used for different environments, e.g. “trafficops_test”, “trafficops”, etc. Many environments choose to use `traffic_ops`.

**description** An optional, human friendly description of the database. Generally this should just describe the purpose of the database e.g. “This database is used for integration testing with our toolset”.

**hostname** The hostname (FQDN) of the server that runs the Traffic Ops Database.

**password** The password to use when authenticating with the Traffic Ops database. In a typical install process, the postinstall script will ask for a password to use for this connection, and this should match that.

**port** The port number (as a string) on which the Traffic Ops Database is listening for incoming connections. traffic_ops_golang ignores this and always uses the default PostgreSQL port (5432).

**ssl** A boolean that sets whether or not the Traffic Ops Database encrypts its connections with SSL.

**type** A string that gives the “type” of database pointed to by all the other options. Once upon a time it was possible for this to either be “mysql” or “postgres”, but the only valid value anymore is “postgres” - and traffic_ops_golang ignores this field entirely (and in fact doesn’t even care if it’s defined at all) and only supports “postgres” databases.
**user**  The name of the user as whom to connect to the database. In a typical install process, the postinstall script will ask for the name of a user to set up for the Traffic Ops Database, and this should match that. Many environments choose to use `traffic_ops`.

**Example database.conf**

```json
{
    "description": "Local PostgreSQL database on port 5432",
    "dbname": "traffic_ops",
    "hostname": "localhost",
    "user": "traffic_ops",
    "password": "password",
    "port": "5432",
    "ssl": false,
    "type": "Pg"
}
```

**influxdb.conf**

This file deals with configuration of the InfluxDB cluster that serves Traffic Stats; specifically it tells Traffic Ops how to authenticate with the InfluxDB cluster and which measurements to check. `traffic_ops_golang` will look for this file at the path given by the value of `influx_db_conf_path` in `cdn.conf`, while the Legacy Perl Script always looks for it at `/directory where cdn.conf is located/value of $MOJO_MODE/influxdb.conf` where the value of `$MOJO_MODE` part is literally the value of the `MOJO_MODE` environment variable. This file is encoded as a JSON object, and its keys are described below.

**See also:**

For more information about InfluxDB, see the InfluxDB documentation.

**cache_stats_db_name**  This field sets the name of the “database” (measurement) used to query for Cache Group statistics. The Legacy Perl Script demands that this be defined, but `traffic_ops_golang` will default to "cache_stats" if this field is not defined. For this reason, it is recommended that this field not be defined if and only if the Traffic Ops server is only running `traffic_ops_golang` (the Legacy Perl Script being run on a different server).

**Danger:** The only valid value for this is "cache_stats", if it is anything else Traffic Stats data for Cache Group statistics will be inaccessible through the Traffic Ops API.

**deliveryservice_stats_db_name**  This field sets the name of the “database” (measurement) used to query for Delivery Service statistics. The Legacy Perl
Script demands that this be defined, but `traffic_ops_golang` will default to "deliveryservice_stats" if this field is not defined. For this reason, it is recommended that this field not be defined if and only if the Traffic Ops server is only running `traffic_ops_golang` (the Legacy Perl Script being run on a different server).

**Danger:** The only valid value for this is "deliveryservice_stats", if it is anything else Traffic Stats data for Delivery Service statistics will be inaccessible through the Traffic Ops API.

**password** Sets the password to use when authenticating with InfluxDB clusters.

**secure** An optional boolean that sets whether or not to use SSL encrypted connections to the InfluxDB cluster (the InfluxDB servers would need to be configured to use SSL). Default if not specified is `false`. Only `traffic_ops_golang` supports SSL for these connections, the Legacy Perl Script will always use insecure connections.

**user** Sets the user name as whom to authenticate with InfluxDB clusters.

**Example influxdb.conf**

```json
{
  "user": "influxuser",
  "password": "password",
  "deliveryservice_stats_db_name": "deliveryservice_stats",
  "cache_stats_db_name": "cache_stats",
  "secure": false
}
```

**ldap.conf**

This file defines methods of connection to an LDAP (Lightweight Directory Access Protocol) server and semantics for searching for users on it for the purpose of authentication. `traffic_ops_golang` will look for this file at the path given by the value of `ldap_conf_location` in `cdn.conf`, while the Legacy Perl Script always looks for it at `/directory where cdn.conf is located/ldap.conf`. ldap.conf’s contents are a JSON-encoded object, the keys of which are detailed below.

**See also:**

For more information on LDAP see the LDAP Wikipedia page and [RFC 4511](https://datatracker.ietf.org/doc/html/rfc4511).

- **admin_dn** The LDAP DN (Distinguished Name) of the administrative user.
- **admin_pass** The password of the administrative user for the LDAP.
**host** The full hostname of the LDAP server, preceded by a scheme (only `ldap://` and `ldaps://` are supported), optionally including port number.

**insecure** A boolean that tells Traffic Ops whether or not to verify the certificate chain of the LDAP server if it uses TLS-encrypted communications.

**ldap_timeout_secs** Sets a timeout in seconds for connections to the LDAP.

**search_base** The directory relative to which searches for users should be conducted.

**search_query** A query to be used to search for users. The string `%s` should appear exactly once in this string, where user names will be inserted procedurally by the handler for LDAP logins.

### Example ldap.conf

```json
{
    "admin_pass" : "password",
    "search_base" : "dc=prefix,dc=domain,dc=suffix",
    "admin_dn" : "user@prefix.domain.suffix",
    "host" : "ldaps://host:[port]",
    "search_query" : "(&
                    (objectCategory=person)
                    (objectClass=user)
                    (userName=%s))",
    "insecure" : false,
    "ldap_timeout_secs" : 20
}
```

### riak.conf

This file sets authentication options for connections to Traffic Vault. `traffic_ops_golang` will look for this file at the path given by the value of the `--riakcfg` flag as passed on startup, while the *Legacy Perl Script* always looks for it at `/directory where cdn.conf is located/value of $MOJO_MODE/riak.conf` where the value of `$MOJO_MODE` part is literally the value of the `MOJO_MODE` environment variable. The contents of `riak.conf` are encoded as a JSON object, the keys of which are described below.

#### Implementation Detail

The name of this file is derived from the current database used in the implementation of Traffic Vault - Riak KV.

- **password** The password to use when authenticating with Traffic Vault
- **user** The username to use when authenticating with Traffic Vault
Example riak.conf

```json
{
    "user": "riakuser",
    "password": "password"
}
```

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)

# 26: Example Procedure

```bash
$ 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>
```

(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.

# 27: Example Procedure

```
$ 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
$ sudo cp localhost.crt /etc/pki/tls/certs
$ sudo cp localhost.key /etc/pki/tls/private
$ sudo chown trafops:trafops /etc/pki/tls/certs=localhost.crt
$ sudo chown trafops:trafops /etc/pki/tls/private/localhost.key
```

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

# 28: Example Certificate Signing Request File Generation

```
$ 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
```

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>
```

(continues on next page)
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

# 29: 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 cdn.conf with any necessary changes.

# 30: 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'
  ...

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:

```
# 31: 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 server 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(8) 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 cache
servers 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 . and domain_name from the server table) ping-able?

RTR (Responds to Traffic Router) Checks the state of each cache server as perceived by all Traffic Monitors (via Traffic Router). This extension asks each Traffic Router for the state of the cache server. A check failure is indicated if one or more monitors report an error for a cache server. A cache server is only marked as good if all reports are positive.

Note: This is a pessimistic approach, opposite of how Traffic Monitor marks a cache server as up, i.e. “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.

# 32: Example Cron File

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

(continues on next page)
*/15 * * * * root /opt/traffic_ops/app/bin/checks/ToPingCheck.pl -c '{"base_url": "https://localhost", "check_name": "10G", "name": "IPv4 Ping", "select": "ipAddress": "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": "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
45 0 * * * root /opt/traffic_ops/app/bin/checks/ToPingCheck.pl -c '{"base_url": "https://localhost", "check_name": "MTU", "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", "name": "Max Trans Unit", "select": "ipAddress", "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", "name": "Max Trans Unit", "select": "ip6Address", "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", "name": "DNS Lookup", "syslog_facility": "local0"]}' > null 2>&1

(continues on next page)
# 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\": ":term:`Delivery Service`\", "cms_interface\": "eth0\", \n  "syslog_facility\": "local0\"\" > /dev/null 2>&1

# RTR

10 * * * * root /opt/traffic_ops/app/bin/checks/ToRTRCheck.pl -c "{\n  "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\", "syslog_facility\": "local0\n  \"\"}" > /dev/null 2>&1

# CHR

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

# ORT

40 * * * * ssh_key_edge_user /opt/traffic_ops/app/bin/checks/ \
  ToORTCheck.pl -c "{\"base_url\": "https://localhost\", "check_ 
  name\": "ORT\"}" >> /var/log/traffic_ops/extensionCheck.log 2>&1

40 * * * * ssh_key_edge_user /opt/traffic_ops/app/bin/checks/ \
  ToORTCheck.pl -c "{\"base_url\": "https://localhost\", "check_ 
  name\": "ORT\", "name\": "Operational Readiness Test\", \n  "syslog_facility\": "local0\"\"}" > /dev/null 2>&1

### 3.1.2 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* if it happens to be in your available repositories (at version 6.0+), or using the NodeSource setup script.

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

4. Install the Traffic Portal RPM with *yum* or *rpm* e.g. by running `yum install path/to/traffic_portal.rpm` as the root user or with `sudo`.

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`)

Configuring OAuth Through Traffic Portal

See Configure OAuth Login.

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`.

```bash
systemctl start traffic_portal
```
Stopping Traffic Portal

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

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

3.1.3 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**

Change Logs

At the top-right of every page is a bubble icon and badge count indicating the number of changes made to the CDN since the last viewing. Clicking on this expands a short list, with an option to \textit{See All Change Logs}. Clicking on this will navigate to the “Changelog” page.

![Changelog Dialog](image)

**Fig. 4: The Changelog Dialog**

Sidebar Bottom Buttons

At the bottom of the sidebar, a few action buttons may be found. At the far left, is the \textit{Logout} button.
Clicking on this button will cause the current user session to end; further use of the Traffic Portal UI will require re-authentication. Next to this is the User Profile button.

Clicking on this button will take the user to his or her user profile page, where much of their information (contact, username etc.) is editable. The next button to the right is the Release Info button.

When this button is clicked, a modal window will appear, displaying the Traffic Portal version, the date on which it was built, and the network location of the Traffic Ops server that serves the Traffic Ops API used by the Traffic Portal instance. The final button is the Popout button.

This button will open the current view in a new browsing context. Most browsers will implement this as a new tab, but can be configured instead to open them in a new window - check your browser’s configuration and set it as desired.
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 *cache servers* across all CDNs. Click the link to view the healthy caches on the cache stats page.

**Unhealthy Caches**  Displays the number of unhealthy *cache servers* 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/app/src/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 TLD (Top-Level Domain)
- **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 Updates on all servers in a CDN, or clear such updates
- Compare CDN Snapshots
- create a CDN Snapshot
- manage a CDN’s DNSSEC keys
- manage a CDN’s Federations
- view Delivery Services of a CDN
- 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. 10: The ‘Monitor’ Menu
Cache Checks

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

**Warning:** Several of these columns may be empty by default - particularly in the CDN in a Box environment - and require Traffic Ops Extensions to be installed/enabled/configured in order to work.

- **Hostname** The (short) hostname of the cache server
- **Profile** The Name of the Profile used by the cache server
- **Status** The Status of the cache server

See also:

*Health Protocol*

- **UPD** Displays whether or not this cache server has configuration updates pending
- **RVL** Displays whether or not this cache server (or one or more of its parents) has content invalidation requests pending
- **ILO** Indicates the status of an ILO (Integrated Lights-Out) interface for this cache server
- **10G** Indicates whether or not the IPv4 address of this cache server is reachable via ICMP “pings”
- **FQDN** DNS check that matches what the DNS servers respond with compared to what Traffic Ops has configured
- **DSCP** Checks the DSCP value of packets received from this cache server
- **10G6** Indicates whether or not the IPv6 address of this cache server is reachable via ICMP “pings”
- **MTU** Checks the MTU by sending ICMP “pings” from the Traffic Ops server
- **RTR** Checks the reachability of the cache server from the CDN’s configured Traffic Routers
- **CHR** Cache-Hit Ratio (percent)
- **CDU** Total Cache-Disk Usage (percent)
- **ORT** Uses the ORT script on the cache server to determine if the configuration in Traffic Ops matches the configuration on cache server itself. The user as whom this script runs must have an SSH key on each server.
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.

Profile Name of the Profile applied to the Edge-tier or Mid-tier cache server, or the special name “ALL” indicating that this row is a group of all cache servers within a single Cache Group.

Host ‘ALL’ for entries grouped by Cache Group, or the hostname of a particular cache server.

Cache Group 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, or the special name “ALL” indicating that this row is an aggregate across all Cache Groups.

Healthy True/False as determined by Traffic Monitor.

See also:

Health Protocol.

Status Status of the cache server or Cache Group.

Connections Number of currently open 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 Delivery Service Requests for such changes - for users without necessary permissions.

3.1. Administrator's Guide

Fig. 11: The ‘Services’ Menu
Delivery Services

This page contains a table displaying all Delivery Services visible to the user as determined by their Tenant.

![Table of Delivery Services](image)

Fig. 12: Table of Delivery Services

Use the Select Columns menu to select the delivery service columns to view and search. Columns can also be rearranged using drag-and-drop. Available delivery service columns include:

- **Active** (visible by default)
- **Anonymous Blocking**
- **CDN** (visible by default)
- **Check Path**
- **Consistent Hashing Query Parameters**
- **Consistent Hashing Regular Expression**
- **Deep Caching**
- **Display Name**
- **DNS Bypass CNAME**
- **DNS Bypass IP**
- **DNS Bypass IPv6**
- **DNS Bypass TTL**
- **DNS TTL**
- **DSCP** (visible by default)
- **Edge Header Rewrite Rules**
- **Fair-Queuing Pacing Rate Bps**
- **Geo Limit**
- **Geo Limit Countries**
- **Geo Limit Redirect URL**
- **Geolocation Provider**
- **Geo Miss Default Latitude**
- **Geo Miss Default Longitude**
• Global Max Mbps
• Global Max TPS
• HTTP Bypass FQDN
• Info URL
• Initial Dispersion
• IPv6 Routing Enabled (visible by default)
• Long Description
• Long Description 2
• Long Description 3
• Max DNS Answers
• Max Origin Connections
• Mid Header Rewrite Rules
• Origin Shield
• Origin Server Base URL (visible by default)
• Profile
• Protocol (visible by default)
• Query String Handling (visible by default)
• Range Request Handling
• Raw Remap Text
• Regex Remap Expression
• Regional Geoblocking
• Routing Name
• Signing Algorithm (visible by default)
• Tenant (visible by default)
• Traffic Router Additional Response Headers
• Traffic Router Log Request Headers
• Type (visible by default)
• Use Multi-Site Origin Feature
• xml_id (visible by default)

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
• Manage *Delivery Service* invalidation requests
• Manage *Delivery Service* origins
• Manage *Delivery Service* regular expressions
• Manage *delivery service required server capabilities*
• Manage *Delivery Service* server assignments
• Manage *Delivery Service* steering targets
• Manage *Delivery Service* static DNS records within a *Delivery Service* subdomain
• Test *Consistent Hashing Patterns*

**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 Requests Table](image)

**Fig. 13: Table of Delivery Service Requests**

*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 configurable table of all servers (of all kinds) across all Delivery Services and CDNs visible to the user.

Use the Select Columns menu to select the server columns to view and search. Columns can also be rearranged using drag-and-drop. Available server columns include:

- **Cache Group** [Visible by default] The Name of the Cache Group to which this server belongs
- **CDN** [Visible by default] The name of the CDN to which the server belongs
- **Domain** [Visible by default] The domain part of the server’s FQDN
- **Host** [Visible by default] The (short) hostname of the server
- **HTTPS Port** The port on which the server listens for incoming HTTPS connections/requests
- **ID** An integral, unique identifier for this server
**ILO IP Address** [Visible by default] The IPv4 address of the server’s ILO service

*See also:*

Hewlett Packard ILO Wikipedia Page

**ILO IP Gateway** The IPv4 gateway address of the server’s ILO service

**ILO IP Netmask** The IPv4 subnet mask of the server’s ILO service

**ILO Username** The user name for the server’s ILO service

**Interface Name** The name of the primary network interface used by the server

**IPv6 Address** [Visible by default] The IPv6 address and subnet mask of `interfaceName`

**IPv6 Gateway** The IPv6 address of the gateway used by `interfaceName`

**Last Updated** The date and time at which this server description was last modified

**Mgmt IP Address** The IPv4 address of some network interface on the server used for ‘management’

**Mgmt IP Gateway** The IPv4 address of a gateway used by some network interface on the server used for ‘management’

**Mgmt IP Netmask** The IPv4 subnet mask used by some network interface on the server used for ‘management’

**Network Gateway** The IPv4 address of the gateway used by `interfaceName`

**Network IP** [Visible by default] The IPv4 address of `interfaceName`

**Network MTU** The Maximum Transmission Unit (MTU) to configured on `interfaceName`

**Network Subnet** The IPv4 subnet mask used by `interfaceName`

**Offline Reason** A user-entered reason why the server is in ADMIN_DOWN or OFFLINE status

**Phys Location** [Visible by default] The name of the physical location where the server resides

**Profile** [Visible by default] The Name of the Profile used by this server

**Rack** A string indicating “server rack” location

**Reval Pending** [Visible by default] A boolean value represented as a clock (content invalidation/revalidation is pending) or green check mark (content invalidation/revalidation is not pending)

**Router Hostname** The human-readable name of the router responsible for reaching this server

**Router Port Name** The human-readable name of the port used by the router responsible for reaching this server
Status [Visible by default] The Status of the server

See also:

Health Protocol

TCP Port The port on which this server listens for incoming TCP connections

Type [Visible by default] The name of the Type of this server

Update Pending [Visible by default] A boolean value represented as a clock (updates are pending) or green check mark (updates are not pending), typically to be acted upon by Traffic Ops ORT

Server management includes the ability to (where applicable):

• Create a new server
• Update an existing server
• Delete an existing server
• Queue Updates on a server, or clear such updates
• Update server status
• View server Delivery Services
• View server configuration files
• Clone Delivery Service assignments
• Assign Delivery Services to server(s)
• Manage server capabilities assigned to a server

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”.

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  The Routing Disabled setting of this Profile

Description  This Profile’s Description

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

3.1. Administrator’s Guide  121
- export a Profile
- view Profile Parameters
- view Profile Delivery Services
- view Profile servers

**Parameters**

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

- **Name** The Name of the Parameter
- **Config File** The Config File to which the Parameter belongs.
- **Value** The Value of the Parameter.
- **Secure** Whether or not the Parameter is Secure
- **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.

![Fig. 16: ‘Topology’ Menu](image)

**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 Updates for all servers in a Cache Group, or clear such updates
- view Cache Group ASNs

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)
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.

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 servers* 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.

See also:

For instructions on setting up the Kickstart ISO generation files, see Creating the CentOS Kickstart File.

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.json 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
   IPv6 Address The resultant system’s IPv6 address
   Network Subnet The system’s network subnet mask
   Network Gateway The system’s network gateway’s IPv4 address
   IPv6 Gateway The system’s network gateway’s IPv6 address
   Management IP Address An optional IP address (IPv4 or IPv6) of a “management” server for the resultant system (e.g. for ILO)
   Management IP Netmask The subnet mask (IPv4 or IPv6) used by a “management” server for the resultant system (e.g. for ILO) - only needed if the Management IP Address is provided
   Management IP Gateway The IP address (IPv4 or IPv6) of the network gateway used by a “management” server for the resultant system (e.g. for ILO) - only needed if the Management IP Address is provided
   Management Interface The network interface used by a “management” server for the resultant system (e.g. for ILO) - only needed if the Management IP Address is provided. Must not be the same as “Interface Name”.

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:
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 hashed using MD5 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 from 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’.

Implementation Detail
Traffic Ops uses Red Hat’s Kickstart <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/installation_guide/chap-kickstart-installations> to create these ISOs, so many configuration options not available here can be tweaked in the Kickstart configuration file.

User Admin

This section offers administrative functionality for users and their permissions.

Fig. 18: 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

**Note:** If OAuth is enabled, the username must exist both here as well as with the OAuth provider. A user’s rights are defined by the role assigned to the user in Traffic Ops. Creating/deleting a user here will update the user’s role but the user needs to be created/deleted with the OAuth provider as well.

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.

![Fig. 19: The ‘Other’ Menu](image)

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.
3.1.4 Traffic Monitor Administration

Installing Traffic Monitor

The following are hard requirements 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 servers. Once every cache server has been polled, Health Protocol state is available via RESTful JSON endpoints and a web browser UI.

Peering and Optimistic Quorum

As mentioned in the Health Protocol section of the Traffic Monitor overview, peering a Traffic Monitor with one or more other Traffic Monitors enables the optimistic health protocol. In order to leverage the optimistic quorum feature along with the optimistic health protocol, a minimum of three Traffic Monitors are required. The optimistic quorum feature allows a Traffic Monitor to withdraw itself from the optimistic health protocol when it loses connectivity to a number of its peers.

To enable the optimistic quorum feature, the peer_optimistic_quorum_min property in traffic_monitor.cfg should be configured with a value greater than zero that specifies the minimum number of peers that must be available in order to participate in the optimistic health protocol. If at any time the number of available peers falls below this threshold, the local Traffic Monitor will serve 503s whenever the aggregated, optimistic health protocol enabled view of the CDN’s health is requested. Traffic Monitor will continue serving 503s and logging errors in traffic_monitor.log until the minimum number of peers are available. Once the minimum number of peers are available, the local Traffic Monitor can resume participation in the optimistic health protocol. This prevents negative states caused by network isolation of a Traffic Monitor from propagating to downstream components such as Traffic Router.

Cache Polling URL

The cache servers are polled at the URL specified in the health.polling.url parameter, on the cache server’s profile.

This parameter must have the config file rascal.properties.

The value is a template with the text ${hostname} being replaced with the cache server’s Network IP (IPv4), and ${interface_name} being replaced with the cache server’s network Interface Name. For example, http://${hostname}/_astats?application=&inf.name=${interface_name}.

If the template contains a port, that port will be used, and the cache server’s HTTPS and TCP Ports will not be added.

If the template does not contain a port, then if the template starts with https the cache server’s HTTPS Port will be added, and if the template doesn’t start with https the cache server’s TCP Port will be added.

Examples:

2.42:8443/_astats?application=&inf.name=${interface_name}.
Template http://${hostname}:1234/_astats?application=&inf.name=${interface_name} Server IP 192.0.2.42 Server TCP Port 8080 HTTPS Port 8443 becomes http://192.0.2.42:1234/_astats?application=&inf.name=${interface_name}.
Template https://${hostname}:1234/_astats?application=&inf.name=${interface_name} Server IP 192.0.2.42 Server TCP Port 8080 HTTPS Port 8443 becomes https://192.0.2.42:1234/_astats?application=&inf.name=${interface_name}.

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 servers 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 servers 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/.

3.1.5 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 Configuration File Parameters

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

# 36: 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)
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27109
;; flags: qr aa rd; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

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

;; 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)
```

(continues on next page)
7. Perform a CDN **Snapshot**.

**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 it 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.
### Table 42: Traffic Router Configuration File Parameters

<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.properties</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.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>
<td></td>
</tr>
<tr>
<td>traffic_monitor.properties</td>
<td>The interval in milliseconds for Traffic Router to poll between waiting for changes to this configuration file</td>
<td>60000</td>
<td></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.tcp.backlog</td>
<td>Maximum length of the queue for incoming TCP connection requests</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>dns.udp.port</td>
<td>UDP port that Traffic Router will use for incoming DNS requests</td>
<td>53</td>
<td></td>
</tr>
<tr>
<td>dns.max-threads</td>
<td>Maximum number of threads used to process incoming DNS requests</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>dns.queue-depth</td>
<td>Maximum number of threads allowed to queue when all workers threads are busy. To disable the queue, set to 0, or to allow an unlimited sized queue, set to -1.</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>dns.zones.dir</td>
<td>Path to automatically generated zone files for reference</td>
<td>/opt/traffic_router/var/auto-zones</td>
<td></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>Full path to 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.geolocation.database.refresh.period</td>
<td>The interval in milliseconds for Traffic Router to wait between polling for changes to the GeoIP2 database</td>
<td>604800000</td>
<td></td>
</tr>
<tr>
<td>cache.czmap.database</td>
<td>Path to the local copy of the coverage zone file</td>
<td>/opt/traffic_router/db/czmap.json</td>
<td></td>
</tr>
<tr>
<td>cache.czmap.database.refresh.period</td>
<td>The interval in milliseconds for Traffic Router to wait between polling for a new coverage zone file</td>
<td>10800000</td>
<td></td>
</tr>
<tr>
<td>cache.szmap.database</td>
<td>The interval in milliseconds for Traffic Router to wait between polling for a new coverage zone file</td>
<td>/opt/traffic_router/db/szmap.json</td>
<td></td>
</tr>
</tbody>
</table>

---

### 3.1. Administrator's Guide
The Traffic Router Profile

Much of a Traffic Router’s configuration can be obtained through the Parameters on its Profile. The Parameters of a Traffic Router’s Profile that have meaning (others are just ignored) are detailed in the The Parameters of a Traffic Router Profile.

<table>
<thead>
<tr>
<th>Name</th>
<th>Config File</th>
<th>Value Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>location</td>
<td>dns.zone</td>
<td>Location to store the DNS zone files in the local file system of Traffic Router.</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 log4j.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) where a Coverage Zone Map may be found.</td>
</tr>
<tr>
<td>ecsEnable</td>
<td>CRConfig.json</td>
<td>Boolean value to enable or disable EDNS0 Client Subnet Extensions.</td>
</tr>
<tr>
<td>geolocation.polling.url</td>
<td>CRConfig.json</td>
<td>The location (URL) where a geolocation database may be found.</td>
</tr>
<tr>
<td>geolocation.polling.interval</td>
<td>CRConfig.json</td>
<td>How often - in milliseconds - Traffic Router should check for an updated geographic IP mapping database.</td>
</tr>
<tr>
<td>coveragezone.polling.interval</td>
<td>CRConfig.json</td>
<td>How often - in milliseconds - Traffic Router should check for an updated geographic IP mapping database.</td>
</tr>
<tr>
<td>coveragezone.polling.url</td>
<td>CRConfig.json</td>
<td>The location (URL) where a Deep Coverage Zone Map may be found.</td>
</tr>
<tr>
<td>deepcoveragezone.polling.interval</td>
<td>CRConfig.json</td>
<td>The location (URL) where a Deep Coverage Zone Map may be found.</td>
</tr>
<tr>
<td>client.steering.forced.diversity</td>
<td>CRConfig.json</td>
<td>When this Parameter exists and is exactly “true”, it enables the “Client Steering Forced Diversity” feature to diversify CLIENT_STEERING results by including more unique Edge-Tier Cache Servers.</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 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 administrative record.</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 value for the “refresh” field the Traffic Router DNS Server will respond with on SOA records.</td>
</tr>
<tr>
<td>tld.ttlNS</td>
<td>CRConfig.json</td>
<td>The TTL the Traffic Router DNS Server will respond with on A records.</td>
</tr>
<tr>
<td>tld.ttlSOA</td>
<td>CRConfig.json</td>
<td>The TTL the Traffic Router DNS Server will respond with on SOA records.</td>
</tr>
<tr>
<td>tld.ttlAAAA</td>
<td>CRConfig.json</td>
<td>The TTL the Traffic Router DNS Server will respond with on AAAA records.</td>
</tr>
<tr>
<td>tld.ttlDNSKEY</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 on which Traffic Router is listening.</td>
</tr>
<tr>
<td>api.cache-control.max-age</td>
<td>CRConfig.json</td>
<td>The value of the Cache-Control: (expires) field in the response to the client’s request.</td>
</tr>
<tr>
<td>api.auth.url</td>
<td>CRConfig.json</td>
<td>The URL of the authentication endpoint.</td>
</tr>
<tr>
<td>consistent.dns.routing</td>
<td>CRConfig.json</td>
<td>Control whether DNS-routed Delivery Services may be used.</td>
</tr>
<tr>
<td>dnssec.enabled</td>
<td>CRConfig.json</td>
<td>Whether DNSSEC is enabled; this parameter enables or disables DNSSEC for the CDN.</td>
</tr>
<tr>
<td>dnssec.allow.expired.keys</td>
<td>CRConfig.json</td>
<td>Allow Traffic Router to use expired DNSSEC keys.</td>
</tr>
<tr>
<td>dynamic.cache.primer.enabled</td>
<td>CRConfig.json</td>
<td>Allow Traffic Router to attempt to prime DNSSEC zones.</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 waits before attempting to prime DNSSEC keys.</td>
</tr>
<tr>
<td>keystore.api.url</td>
<td>CRConfig.json</td>
<td>The URL of the DNSSEC key management API.</td>
</tr>
<tr>
<td>keystore.fetch.timeout</td>
<td>CRConfig.json</td>
<td>The timeout in milliseconds for requests to the DNSSEC key management API.</td>
</tr>
<tr>
<td>keystore.fetch.retries</td>
<td>CRConfig.json</td>
<td>The number of times Traffic Router will wait for requests to the DNSSEC key management API.</td>
</tr>
<tr>
<td>Name</td>
<td>Config File</td>
<td>Value</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>keystore.fetch.wait</td>
<td>CRConfig.json</td>
<td></td>
</tr>
<tr>
<td>signaturemanager.expiration.multiplier</td>
<td>CRConfig.json</td>
<td></td>
</tr>
<tr>
<td>zonemanager.threadpool.scale</td>
<td>CRConfig.json</td>
<td></td>
</tr>
<tr>
<td>zonemanager.cache.maintenance.interval</td>
<td>CRConfig.json</td>
<td></td>
</tr>
<tr>
<td>zonemanager.dynamic.response.expiration</td>
<td>CRConfig.json</td>
<td></td>
</tr>
<tr>
<td>zonemanager.dynamic.concurrentylevel</td>
<td>CRConfig.json</td>
<td></td>
</tr>
<tr>
<td>zonemanager.dynamic.init.capacity</td>
<td>CRConfig.json</td>
<td></td>
</tr>
<tr>
<td>zonemanager.init.timeout</td>
<td>CRConfig.json</td>
<td></td>
</tr>
<tr>
<td>DNSKEY.generation.multiplier</td>
<td>CRConfig.json</td>
<td></td>
</tr>
<tr>
<td>DNSKEY.effective.multiplier</td>
<td>CRConfig.json</td>
<td></td>
</tr>
</tbody>
</table>

Deprecated since version ATCv4.0: The use of “CRConfig.xml” as a Parameter “Config File” value has no known meaning, and its use for configuring Traffic Router is deprecated. All configuration (?) that previously used that value should instead use the equivalent Parameter with the Config File value “CRConfig.json”.

### Consistent Hashing

Traffic Router does special optimization for some requests to ensure that requests for specific content are consistently fetched from a small number (often exactly one, but dependent on Initial Dispersion) of cache servers - thus ensuring it stays “fresh” in the cache. This is done by performing “consistent hashing” on request paths (when HTTP routing) or names requested for resolution (when DNS routing). To an extent, this behavior is configurable by modifying fields on Delivery Services. Consistent hashing acts differently on a Delivery Service based on how Delivery Services of its Type route content.

- **HTTP, HTTP_NO_CACHE, HTTP_LIVE, HTTP_LIVE_NATNL, DNS, DNS_LIVE, and DNS_NATNL** These Delivery Service Types route directly to cache servers, so consistent hashing is used to choose a cache server to which the client will be redirected.

- **STEERING and CLIENT_STEERING** These Delivery Service Types route to “target” Delivery Services, so consistent hashing is used to choose a “target” which will service the client request.

See also:

See the Wikipedia article on consistent hashing.

### Consistent Hashing Patterns

New in version 4.0.

Regular expressions (“patterns”) can be provided in the Consistent Hashing Regular Expression field of an HTTP-routed Delivery Service to influence what parts of an HTTP request path are considered when performing consistent hashing. These patterns propagate to Traffic Router through Snapshots.
Important: Consistent Hashing Patterns on STEERING-Type Delivery Services will be used for Consistent Hashing - the Consistent Hashing Pattern(s) of said Delivery Service’s target(s) will not be considered. If Consistent Hashing Patterns are important to the routing of content on a STEERING-Type or CLIENT_STEERING-Type Delivery Service, they must be defined on that Delivery Service itself, and not on its target(s).

How it Works

The supplied Consistent Hashing Regular Expression is applied to the request path to extract matching elements to build a new string before consistent hashing is done. For example, using the pattern /.*?(/.*?/).*?(m3u8) and given the request paths /test/path/asset.m3u8 and /other/path/asset.m3u8 the resulting string used for consistent hashing will be /path/m3u8

See also:

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

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 exist to test regular expression application against a request path, cache server selection, and Delivery Service selection.

- **Traffic Ops API** The consistenthash endpoint will proxy request data through to one of the Traffic Router endpoints in order to test regular expression application against a request path, in the event that direct access to the Traffic Router API is not possible and/or desired.

- **Traffic Portal** On the Delivery Service creation/modification form in Traffic Portal (under Delivery Services), there is a Test Regex section that the user can use to validate a regular expression before saving it to a Delivery Service.

Consistent Hash Query Parameters

Normally, when performing consistent hashing for an HTTP-routed Delivery Service, any query parameters present in the request are ignored. That is, if a client requests /some/path?key=value consistent hashing is only performed on the string ‘/some/path’. However, query parameters that are part of uniquely identifying content can be specified by adding them to the set of Consistent Hashing Query Parameters of a Delivery Service. For example, suppose that the file /video.mp4 is available on the origin server in different resolutions, which are specified by the resolution query parameter. This means that /video.mp4?resolution=480p and /video.mp4?resolution=720p share a request path,
but represent different content. In that case, adding resolution to the Delivery Service’s Consistent Hashing Query Parameters will cause consistent hashing to be done on e.g. /video.mp4?resolution=480p instead of just /video.mp4 - however if the client requests e.g. /video.mp4?resolution=480p&bitrate=120kbps consistent hashing will only consider /video.mp4?resolution=480p.

Note: Consistent Hashing Patterns are applied before query parameters are considered - i.e. a pattern cannot match against query parameters, and need not worry about query parameters contaminating matches.

Important: Consistent Hash Query Parameters on the targets of STEERING-Type Delivery Services will be used for Consistent Hashing - the Consistent Hash Query Parameters of said Delivery Services themselves will not be considered. If Consistent Hash Query Parameters are important to the routing of content on a STEERING-Type or CLIENT_STEERING-Type Delivery Service, they must be defined on that Delivery Service’s target(s), and not on the Delivery Service itself.

Caution: Certain query parameters are reserved by Traffic Router for its own use, and thus cannot be present in any Consistent Hash Query Parameters. These reserved parameters are:
- trred
- format
- fakeClientIPAddress

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 KSK (Key Signing Key)s and ZSK (Zone Signing Key)s 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 in The Traffic Router Profile.

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.: demo1.mycdn.ciab.test) for every valid key that exists, in addition to the CDN TLD’s zone. 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 \texttt{/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 \texttt{unquoted_string=\text"optionally quoted string"}
- Values that are quoted strings may contain whitespace characters.
- Values that are not quoted should not contain any whitespace characters.

\textbf{Note:} Any value that is a single dash character or a dash character enclosed in quotes represents an empty value.
### Table 44: 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

```
# 37: 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

144 Chapter 3. Administrator's Guide
**rptye 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 (through the edgeLocation field of a Snapshot) or via coordinates (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

NO_DETAILS  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

# 38: 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=GETcqhv=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 45: 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 46: 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

# 39: Sample Message

```
144140678.000 qtype=DNS chi=192.168.10.11 rhi=- ttms=123 xn=65535,
→ fqdn=www.example.com. type=A class=IN ttl=12345 rcode=NOERROR,
→ rtype=CZ rloc="40.252611,58.439389" rdtl=- rerr="-" ans="192.168.
→ 1.2 192.168.3.4 0:0:0:0:ffff:c0a8:102 0:0:0:0:ffff:c0a8:304"
```

Table 47: Request Fields

<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 EDNS0 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 be</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 48: Response Fields

<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 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 servers on a per-Delivery Service basis. The term “deep” is used in the networking
sense, meaning that the Edge-tier *cache servers* 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 *Deep Coverage Zone File* mapping these deep cache hostnames to specific network
prefixes
3. Deep caching *Parameters* in the Traffic Router *Profile*
   - deepcoveragezone.polling.interval
   - deepcoveragezone.polling.url

   **See also:**
   
   See *The Traffic Router Profile* for details.

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 (Deep Coverage Zone File) is preferred over the regular CZF (Coverage Zone
File) 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 servers* in the client’s corre-
sponding zone. If there are no “deep” *cache servers* 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 an-
other *Delivery Service*. The *Type* of a Steering *Delivery Service* is either STEERING or
CLIENT_STEERING. A Steering *Delivery Service* will have target *Delivery Services* config-
ured 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 con-
figured 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.

**Let’s Encrypt**

Let’s Encrypt is a free, automated CA using ACME (Automated Certificate Management Environment) protocol. Let’s Encrypt performs a domain validation before issuing or renewing a certificate. There are several options for domain validation but for this application the DNS challenge is used in order to receive wildcard certificates. Let’s Encrypt sends a token to be used as a TXT record at `_acme-challenge.domain.example.com` and after verifying that the token is accessible there, will return the newly generated and signed certificate and key. The basic workflow implemented is:

1. **POST** to Let’s Encrypt and receive the DNS challenge token.
2. Traffic Ops stores the DNS challenge.
3. Traffic Router has a watcher which checks with Traffic Ops for any new challenges or deleted challenges.
4. When a new record appears, Traffic Router temporarily adds a static route for the specified **Delivery Service** with the token from Let’s Encrypt at `_acme-challenge.domain.example.com`.
5. Let’s Encrypt continuously attempts to resolve it as a TXT record to verify ownership of the domain.

**Note:** DNSSEC should be turned on for any CDN using Let’s Encrypt to guard against a ‘Man in the Middle’ interference with this transaction.

1. Let’s Encrypt returns the signed certificate and key to Traffic Ops.
2. Traffic Ops stores the certificate and key in Traffic Vault and removes the DNS challenge record.
3. The Traffic Router watcher removes the TXT record.

Let’s Encrypt can be set up through `cdn.conf` by updating the following fields:
Table 49: Fields to update for Let’s Encrypt under `lets_encrypt`

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user_email</td>
<td>string</td>
<td>Optional. Email to create account with Let’s Encrypt or to receive expiration updates</td>
</tr>
<tr>
<td>send_expiration_email</td>
<td>boolean</td>
<td>Option to send email summarizing certificate expiration status</td>
</tr>
<tr>
<td>convert_self_signed</td>
<td>boolean</td>
<td>Option to convert self signed to Let’s Encrypt certificates as they expire. Only works for certificates labeled as Self Signed in the Certificate Source field.</td>
</tr>
<tr>
<td>renew_days_before_expiration</td>
<td>int</td>
<td>Number of days before expiration date to renew certificates</td>
</tr>
<tr>
<td>environment</td>
<td>string</td>
<td>Let’s Encrypt environment to use. Options are ’staging’ or ’production’. Defaults to ‘production’</td>
</tr>
</tbody>
</table>

Table 50: Fields to update for sending emails under `smtp`

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enable sending emails through Simple Mail Transfer Protocol (SMTP)</td>
</tr>
<tr>
<td>user</td>
<td>string</td>
<td>User for SMTP server access</td>
</tr>
<tr>
<td>password</td>
<td>string</td>
<td>Password for SMTP server access</td>
</tr>
<tr>
<td>address</td>
<td>string</td>
<td>SMTP server address including port</td>
</tr>
</tbody>
</table>

**Suggested Way of Setting up an HTTPS Delivery Service With Let’s Encrypt Automation**

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. Click on the *Use Let’s Encrypt* slider, click on the green *Generate Keys* button, then confirm that you want to make these changes
5. 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*.
Suggested Way of Setting up an HTTPS Delivery Service With Certificate Authority

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:

- 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`
6. Using your web browser of choice, open the file test/router/index.html
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.

# 40: Example CATALINA_OPTS Configuration

```plaintext
CATALINA_OPTS="\n-server -Xms2g -Xmx8g \n```
(continues on next page)
-Dlog4j.configuration=file://$CATALINA_BASE/conf/log4j.properties \
-Djava.library.path=/usr/lib64 \
-XX:+UseG1GC \
-XX:+UnlockExperimentalVMOptions \
-XX:InitiatingHeapOccupancyPercent=30"

3.1.6 Traffic Router - Migrating to 3.0

Contents

- Traffic Router - Migrating to 3.0
  - Release Notes v3.0
  - System Requirements
  - Upgrade Procedure
  - Development Environment Upgrade

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 systemctl(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 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.

```
# 41: 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
```

3.1.7 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 statistics
- **dailySummaryRetentionPolicy** The retention policy to be used for the daily statistics
- **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 (when not configured, this menu item will not appear). 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.

# 42: Sample Query

```
SELECT sum(value)*1000 FROM "monthly"."bandwidth.cdn.˓
˓1min" GROUP BY time(60s), cdn;
```

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 Javascript files in traffic_stats/grafana/ 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.

See also:

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 a Profile that has the Type InfluxDB. Make sure to use port 8086 in the configuration.
- The traffic stats server should be added to Traffic Ops with a Profile that has the Type 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 special GLOBAL Profile. This assumes you followed instructions in the Installation, Configuring Traffic Stats, Configuring InfluxDB, and Configuring Grafana sections.

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

where

grafanaHost is the FQDN of the Grafana server (again, usually the same as the Traffic Stats server),
cdn-stats-dashboard is the name of the Dashboard providing CDN-level statistics,

deliveryservice-stats-dashboard is the name of the Dashboard providing Delivery Service-level statistics, and
**daily-summary-dashboard** is the name of the Dashboard providing a daily summary of general statistics that would be of interest to administrators using Traffic Portal.

### 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 a 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

3.1.8 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
# 43: Apache Traffic Server Installation Using `yum(8)`

```
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

```
# 44: Starting ATS with `systemd(1)`

```
systemctl start trafficserver
```

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

```
# 45: 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

```
# 46: 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.

ORT Config File Generation

In the past, ATS config files were generated by Traffic Ops. Traffic Control is in the process of moving ATS config file generation to a library for generic use, and to an application which uses the library and resides on the cache.

The library, lib/go-atscfg, allows users to write their own applications and servers, if they wish to generate ATS configuration files and deploy them to caches via other means. For example, if you wish to generate config files with an additional service, or continue generating config files on Traffic Ops itself via a plugin or local service.

The app, atstccfg, is installed by the traffic_ops_ort RPM alongside the ORT script. This app makes standard API calls to Traffic Ops, and uses their data to build the ATS config files. The ORT script now requests all config through the app, which generates config files it has, and requests directly from Traffic Ops the files it doesn’t recognize.

This provides several benefits. Primarily, reduces the overhead and risk of the monolithic Traffic Ops installation and upgrade process, and allows operators to canary-test config changes one cache at a time, and in the event of an error, only rolling back a few canary caches rather than the entire Traffic Ops instance.

In order to see which config files are generated by a given ORT or atstccfg version, run /opt/ort/atstccfg --print-generated-files.

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

   # 47: 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.
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 server* s) 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 server* s. 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.

### 3.1.9 Traffic Vault Administration
Installing Traffic Vault

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

```bash
# 48: 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

```
# 49: 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)
Apache Traffic Control Documentation, Release 4.0.0

(continued from previous 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.

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
# 50: 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

```
# 51: 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.

```
# 52: 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
```

(continues on next page)
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.

```
# 53: 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 -kvsX 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.

```
# 54: 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 -kvsX 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

```
# 55: Using the Riak API to Create an Index-to-Bucket Association for sslkeys

# Note that the assumptions made here are that the "admin" user's password is "pass"
```

(continues on next page)
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.

### Note that the following steps should be done on the Traffic Vault server ###

# Obtain the script - in this example by downloading it from GitHub

# Assuming Traffic Ops is hosted at trafficops.infra.ciab.test, with username 'admin' and password 'twelve!'
# the script should be run like so:
./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/sslkeys

### Validate Riak Search is Working ###

# Note that the assumptions made here are that the "admin" user's password is
"pass", the Traffic Vault server's Riak database is accessible at port 8088 on
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
```bash
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
```bash
```

### 3.1.10 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.

```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"],

    "ip6Whitelist": ["2001:550:90a::/48", "::1/128",
                     "::1/128"],

    "redirectUrl": "http://youvebeenblocked.com"
}
```

(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 Networks), 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 with the “CR-Config.json” Config File, and ensure they are assigned to all of the Traffic Routers that should perform Anonymous Blocking:

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

   anonymousip.polling.url The 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 using the Delivery Services view in Traffic Portal (don’t forget to save changes!)
4. Go to the Traffic Portal CDNs view, click on Diff CDN Config Snapshot, and click Perform Snapshot.

**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.
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.

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.

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

“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 >= 17.05.0-ce
- Docker Compose\(^1\) version >= 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`

\(^1\) 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 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`
• 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 -d` 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 environment are by default not exposed locally to the host. If this is the desired behavior when bringing up CDN in a Box the command `docker-compose -f docker-compose.yml -f docker-compose.expose-ports.yml up` should be run. The ports are configured within the `infrastructure/cdn-in-a-box/docker-compose.expose-ports.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 52: 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 60443³</td>
<td>TO_ADMIN_USER</td>
<td>TO_ADMIN_PASSWORD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>in variables.env</td>
<td>in variables.env</td>
</tr>
<tr>
<td>Traffic Ops PostgresQL Database</td>
<td>PostgreSQL connections accepted on port 5432 (database name: DB_NAME in variables.env)</td>
<td>DB_USER in variables.env</td>
<td>DB_USER_PASS in variables.env</td>
</tr>
<tr>
<td>Traffic Portal</td>
<td>Web interface on 443 (Javascript required)</td>
<td>TO_ADMIN_USER</td>
<td>TO_ADMIN_PASSWORD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>in variables.env</td>
<td>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 (HTTP) and 2222 (HTTPS)</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</td>
<td>TV_ADMIN_PASSWORD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>in variables.env</td>
<td>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>Influxdb connections accepted on port 8086 (database name: cache_stats, daily_stats and deliveryservice_stats)</td>
<td>INFLUXDB_ADMIN_USER</td>
<td>INFLUXDB_ADMIN_PASSWORD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>in variables.env</td>
<td>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](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 nu-

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.
anced purpose not discussed here but already has the `curl(1)` command line tool installed. For a more user-friendly interface into the CDN network, see `VNC Server`.

# 59: Example Command to See the CDN in Action

```
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=jhds1vhdfsuklvfhsuvlhs
TM_USER=tmon
TM_LOG_EVENT=stdout
TM_LOG_ERROR=stdout
TM_LOG_WARNING=stdout
TM_LOG_INFO=stdout
TM_LOG_DEBUG=stdout
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
TO_LOG_ERROR=stdout
TO_LOG_WARNING=stdout
TO_LOG_INFO=stdout
TO_LOG_DEBUG=stdout
TO_LOG_EVENT=stdout
TP_HOST=trafficportal
TP_EMAIL=tp@cdn.example.com
TP_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_uJpkfjfiBmi1319vfgL10
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
ENROLLENDER_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 53: Self-Signed X.509 Certificate List

<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</td>
<td>N/A</td>
</tr>
<tr>
<td>infra.ciab.test.crt</td>
<td>Infrastruture Certificate</td>
<td>prefix.infra.ciab.test</td>
</tr>
<tr>
<td>demo1.mycdn.ciab.test.crt</td>
<td>Demo1 Delivery Service Certificate</td>
<td>prefix.demo1.mycdn.</td>
</tr>
<tr>
<td>demo2.mycdn.ciab.test.crt</td>
<td>Demo2 Delivery Service Certificate</td>
<td>prefix.demo2.mycdn.</td>
</tr>
</tbody>
</table>

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.
Tsonging the Certificate Authority

For developer and testing use-cases, it may be necessary to have full x509 CA validation by HTTPS clients. 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.

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 ‘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 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.
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/`.

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/topology 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. 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 Up-
dates) will be delayed AUTO_SNAPQUEUE_ACTION_WAIT seconds.

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:

---

8 Automatic Snapshot/Queue Updates is enabled by default in variables.env.
9 Server poll interval and delay action wait are defaulted to a value of 2 seconds.
# 60: Starting Optional Containers with an Alias

```bash
# 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 $PWD/
  optional/docker-compose.$NAME1.yml -f $PWD/optional/docker-
  compose.$NAME2.yml"
docker volume prune -f
mydc build
mydc up
```

## 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/varibles.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
# 61: CIAB Startup Using Bash Alias
```

```bash
# From infrastructure/cdn-in-a-box
alias mydc="docker-compose "
  "-f $PWD/docker-compose.yml "
  "-f $PWD/docker-compose.expose-ports.yml "
  "-f $PWD/optional/docker-compose.vnc.yml "
  "-f $PWD/optional/docker-compose.vnc.expose-ports.yml"
docker volume prune -f
mydc build
mydc kill
mydc rm -fv
mydc up
```

4. Connect with a VNC client to localhost port 5909.

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>Whitelist</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

```
# 62: CIAB Startup Using Bash Alias

# 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.

```
# 63: 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.
# 64: 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.

```
# 65: 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`.

```
# 66: Install openvpn on OSX
brew install openvpn
```

If you want a GUI version of VPN client, we recommend [Tunnelblick](https://tunnelblick.net/).

**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`.  

3.1. Administrator's Guide
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.

```
# 67: CIAB Startup with Grafana

# From infrastructure/cdn-in-a-box
alias mydc="docker-compose -f $PWD/docker-compose.yml -f $PWD/
  optional/docker-compose.grafana.yml -f $PWD/optional/docker-
  compose.grafana.expose-ports.yml"
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=`. 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=`. The query parameter which in this particular URL should be the xml_id of the desired Delivery Service.
Forcing Content Invalidation

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.

Implementation Detail

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 Control does not try to remove the content from the caches, but it makes the content inaccessible using the regex_revalidate plugin for Apache Traffic Server. This forces a “re-validation” of the content.

Note: This method forces cache servers to “re-validate” content, so in order to work properly the origin needs to support revalidation according to section 13 of RFC 2616.

To invalidate content for a specific Delivery Service, follow these steps:

1. Select the desired Delivery Service from the Delivery Services view of Traffic Portal

![Fig. 26: The Traffic Portal Delivery Services view](image)

2. From the More drop-down menu, select Manage Invalidation Requests

3. Click/tap on the + button to open the submission form for a new content invalidation. Fill out this form. The “Path Regex” field should be a PCRE-compatible regular expression that matches all content that must be invalidated - and should not match any content that must not be invalidated. “TTL (hours)” specifies the number of hours for which the invalidation should remain active. Best practice is to set this to the same as the content’s cache lifetime (typically set in the origin’s Cache-Control response header).
Fig. 27: Select ‘Manage Invalidation Requests’
Configure DNSSEC

In order to support DNSSEC in Traffic Router, Traffic Portal provides some actions for managing DNSSEC keys for a CDN and associated Delivery Services. DNSSEC keys consist of a KSK which is used to sign other DNSKEY records as well as a ZSK which is 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 API to retrieve this information.

See also:

DNSSEC

Note: DNSSEC is only supported for DNS-Routed Delivery Services.

Creating Keys

Note: In order for Traffic Ops to successfully store keys in Traffic Vault, at least one Traffic Vault server needs to be configured in Traffic Ops.

1. Go to CDNs and click on the desired CDN.
2. Click on More → Manage DNSSEC Keys.

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.

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.

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.

**Enabling and Disabling DNSSEC on a CDN**

1. 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 to the CDN details page for this CDN, and set the ‘DNSSEC Enabled’ field to ‘true’ (or ‘false’ to disable DNSSEC), then click the green *Update* button.
2. DNSSEC should now be active (or inactive, if disabled) 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
```

### Regenerating KSKs

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.

### Regenerating Keys

It’s also possible to re-do the generation of DNSSEC keys once they have already been gen-
erated. This can be done at any point after *Creating Keys* by clicking on the then-enabled
*Regenerate Keys* button on the CDN’s details page.

### 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
configuration updates (i.e. *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** Filling 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 (i.e. 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 (i.e. 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’.

**Note:** Only the user that fulfilled the delivery service request can mark a delivery service as ‘complete’. This prevents other users from interfering in the process and marking delivery services as ‘complete’ when further action is required for the changes to truly be deployed. However, in traffic_portal_properties.json, users with the ‘overrideRole’ are given the ability to mark any delivery service requests as ‘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).

```bash
# 68: Example cURL Command
curl -i -XPOST "http://localhost:3000/api/1.1/user/login" -H "Content-Type: application/json" -d '{ "u": "federation_user1", "p": "password" }'

# 69: Example API Response
HTTP/1.1 200 OK
Date: Wed, 02 Dec 2015 21:12:06 GMT
Content-Length: 65

(continues on next page)
Successfully added Federation!

- **CNAME**: img.mega-cdn.net.
- **Description**: Federation Mapping for img.mega-cdn.
- **TTL**: 30
- **Role**: federation
- **Delivery Services**: images-c1
- **User**: Federation User (federation_user1)

**Resolvers**
6. The federation user sends a request to Traffic Ops to add IPV4 and/or IPV6 resolvers

```bash
# 70: Example cURL Command

curl -ki -H "Cookie: mojolicious=eyJleHBpcmVzIjoxNDQ5MTA1MTI2LCJhdXRoX2RhdGEiOiJmZWRlcmF0aW9uX3VzZXIxIn0--06b4f870d809d82a91433e92eae8320875c3e8b0;" -XPUT 'http://localhost:3000/api/1.2/federations' -d '{"federations": [ { "deliveryService": "images-ci", "mappings": [ { "resolve4": [ "8.8.8.8/32", "8.8.4.4/32" ], "resolve6": ["2001:4860:4860::8888/128", "2001:4860:4860::8844"] } ] }'}
```

# 71: Example API Response

```
HTTP/1.1 200 OK
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Server: Mojolicious (Perl)
Date: Wed, 02 Dec 2015 21:25:42 GMT
Content-Length: 74
Access-Control-Allow-Credentials: true
Content-Type: application/json
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
```

(continues on next page)
7. The resolvers added by the Federation-user will now be 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.
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.
# 74: Creating a New System Image Definition Tree from an Existing One

```
# 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.json
```

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.

**Configure Multi-Site Origin**

The following steps will take you through the procedure of setting up an MSO.

1. Create a profile to assign to each of the origins:
2. Create server entries for the origination vips:

3. Check the multi-site check box in the *Delivery Service* screen:

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
OSBU 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 the following Parameters:
<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Config File Name</th>
<th>Value</th>
<th>ATS parent.config value</th>
</tr>
</thead>
<tbody>
<tr>
<td>mso.algorithm</td>
<td>parent.config</td>
<td>true, false, strict, consistent_hash</td>
<td>round_robin</td>
</tr>
<tr>
<td>mso.parent_retry</td>
<td>parent.config</td>
<td>simple_retry, both, unavailable_server_retry</td>
<td>parent_retry</td>
</tr>
<tr>
<td>mso.unavailable_server_retry_responses</td>
<td>parent.config</td>
<td>list of server response codes, eg “500,502,503”</td>
<td>defaults to the value in records.config when unused.</td>
</tr>
<tr>
<td>mso.max簡單_retries</td>
<td>parent.config</td>
<td>Number of retries made after a 4xx error</td>
<td>defaults to the value in records.config when unused.</td>
</tr>
<tr>
<td>mso.max_unavailable_server_retries</td>
<td>parent.config</td>
<td>Number of retries made after a 5xx error</td>
<td>defaults to the value in records.config when unused.</td>
</tr>
</tbody>
</table>

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 OAuth Login

An opt-in configuration for SSO using OAuth is supported and can be configured through the `/opt/traffic_portal/public/traffic_portal_properties.json` and `/opt/traffic_ops/app/conf/cdn.conf` files. OAuth uses a third party provider to authenticate the user. Once enabled, the Traffic Portal Login page will no longer accept username and password but instead will authenticate using OAuth. This will redirect to the OAuthUrl from `/opt/traffic_portal/public/traffic_portal_properties.json` which will authenticate the user then redirect to the new `/sso` page with an authorization code. The new `/sso` page will then construct the full URL to exchange the authorization code for a JSON Web Token, and POST this information to the `user/login/oauth` API endpoint. The `user/login/oauth` API endpoint will POST to the URL provided and receive JSON Web Token. The `user/login/oauth` API endpoint will decode the token, validate that it is between the issued time and the expiration time, and validate that the public key set URL is allowed by the list of whitelisted URLs read from `/opt/traffic_ops/app/conf/cdn.conf`. It will then authorize the user from the database and return a mojolicious cookie as per the normal login workflow.

**Note:** Ensure that the user names in the Traffic Ops database match the value returned in the `sub` field in the response from the OAuth provider when setting up with the OAuth provider. The `sub` field is used to reference the roles in the Traffic Ops database in order to authorize the user.

**Note:** OAuth providers sometimes do not return the public key set URL but instead require a locally stored key. This functionality is not currently supported and will require further development.

**Note:** The POST from the API to the OAuth provider to exchange the code for a token expects the response to have the token in JSON format with `access_token` as the desired field (and can include other fields). It also supports a response with just the token itself as the body. Further development work will need to be done to allow other response forms or other response fields.

**Note:** Users must exist in both Traffic Ops as well as in the OAuth provider’s system. The user’s rights are defined by the `role` assigned to the user.

To configure OAuth login:

- Set up authentication with a third party OAuth provider.
- Update `/opt/traffic_portal/public/traffic_portal_properties.json` and ensure the following properties are set up correctly:
Table 57: OAuth Configuration Property Definitions In traffic_portal_properties.json

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Allow OAuth SSO login</td>
</tr>
<tr>
<td>OAuthUrl</td>
<td>string</td>
<td>URL to your OAuth provider</td>
</tr>
<tr>
<td>redirectUriParameterOverride</td>
<td>string</td>
<td>Query parameter override if the OAuth provider requires a different key for the redirect_uri parameter, defaults to redirect_uri</td>
</tr>
<tr>
<td>clientId</td>
<td>string</td>
<td>Client id registered with OAuth provider, passed in with client_id parameter</td>
</tr>
<tr>
<td>OAuthCodeTokenUrl</td>
<td>string</td>
<td>URL to your OAuth provider’s endpoint for exchanging the code (from OAuthUrl) for a token</td>
</tr>
</tbody>
</table>

# 75: Example OAuth Configuration Properties In traffic_portal_properties.json

```json
{
    "oAuth": {
        "_comment": "Opt-in OAuth properties for SSO login. See http://traffic-control-cdn.readthedocs.io/en/release-4.0.0/admin/quick_howto/oauth_login.html for more details.
redirectUriParameterOverride defaults to redirect_uri if left blank.",
        "enabled": true,
        "OAuthUrl": "example.oauth.com",
        "redirectUriParameterOverride": "",
        "clientId": "",
        "OAuthCodeTokenUrl": "example.oauth.com/oauth/token"
    }
}
```

- Update /opt/traffic_ops/app/conf/cdn.conf property traffic_ops_golang.whitelisted_oauth_urls to contain all allowed domains for the JSON key set (Use * for wildcard):

Table 58: OAuth Configuration Property Definitions In cdn.conf

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>whitelisted_oauth_urls</td>
<td>array of strings</td>
<td>List of whitelisted URLs for the JSON public key set returned by OAuth provider. Can contain * wildcards.</td>
</tr>
<tr>
<td>oauth_client_secret</td>
<td>string</td>
<td>Client secret registered with OAuth provider to verify client, passed in with client_secret parameter</td>
</tr>
</tbody>
</table>
# 76: Example OAuth Configuration Properties In cdn.conf

```
{
    "traffic_ops_golang": {
        "whitelisted_oauth_urls": [
            "oauth.example.com",
            "*.example.com"
        ],
        "oauth_client_secret": "secret"
    }
}
```

## Compare Profiles

In Traffic Portal all users can compare the Parameters of any 2 Profiles side-by-side, and users with a higher level Role (“operations” or “admin”) can easily add or remove parameters from each profile as necessary.

The ability to compare 2 profiles can be found under Configure → Profiles → More → Compare Profiles.

![Fig. 37: The “Compare Profiles” menu item](image)

Once you have selected the Compare Profiles menu item, you will be asked to choose 2 profiles to compare.

![Fig. 38: The “Compare Profiles” dialog](image)

All parameters exclusively assigned to one profile but not the other will be displayed by default with their profile membership displayed side-by-side. In addition, by selecting the Show Shared Params link, the user can see a superset of all parameters (shared and not shared) across the 2 profiles. Both views provide users with higher level permissions (“operations” or “admin”) the ability to easily remove or add parameters for each profile and persist the final state of both profiles (or restore the original state and discard changes). As the user makes changes, a blue shadow is added to all modified checkboxes.
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",  
        "isSteeringDS": "false"  
      },  
      {  
        "deliveryServiceId": "linear-steering",  
        "urlRegex": ".*live3\./m3u8",  
        "ipWhiteList": ["185.68.71.9/22", "142.232.0.79/24"]  
      }  
  }  
}  
```

(continues on next page)
"geoLocation": {
    "excludePostalCode": ["N0H", "L9V"],
    "redirectUrl": "http://ip-slatecdn.example.com/slate.m3u8",
    "isSteeringDS": "true"
}
}

**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. Steering-Type Delivery Services must contain an FQDN as the re-direct or Traffic Router will return a DENIED to the client. This is because steering services do not have caches associated to them, so a relative redirectURL can not be turned into a FQDN.

**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.

2. Add RGB Parameters in Traffic Portal to the Delivery Service’s Traffic Router(s)’s Profile(s). The Config File value should be set to CRConfig.json, and the following two Parameter Name/Value pairs need to be specified:

   **regional_geoblock.polling_url** The URL of the RGB configuration file. Traffic Router will fetch the file from this URL using an HTTP GET request.
The interval on which Traffic Router polls the RGB configuration file.

3. Enable RGB for a Delivery Service using the Delivery Services view in Traffic Portal (don’t forget to save changes!)

4. Go to the Traffic Portal CDNs view, click on Diff CDN Config Snapshot, and click Perform Snapshot.

Traffic Router Access Log

See also:

Troubleshooting and Log Files

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.

# 78: 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="-"
```

Manage Server Capabilities

Server capabilities are designed to enable users with the Operations or Admin Role to control the flow of delivery service traffic through only the cache servers (Edge or Mid) that have the required capabilities. For example, delivery services designed to serve large binary files should only have requests routed to cache servers with sufficient disk cache. Currently, this can be controlled at the Edge-tier where system operators can explicitly assign only Edge-tier caches with sufficient disk cache to the delivery service. However, operators do not have control of Mid-tier cache assignments and cannot dictate which Mid-tier caches are qualified to serve these large binary files. This will cause a problem if a Mid-tier cache with insufficient disk cache is asked to serve the delivery services large binary files.

A list of the server capabilities can be found under Configure → Server Capabilities. Users with a higher-level role (“operations” or “admin”) can create or delete server capabilities. Server capabilities can only be deleted if they are not currently being used by a cache server or required by a delivery service.
Manage server capabilities assigned to a server

Users with the Operations or Admin Role can associate one or more server capabilities with a cache servers (Edge or Mid) by navigating to a server via Configure → Servers and using the context menu for the server table and selecting Manage Capabilities or by navigating to Configure → Servers → Server → More → Manage Capabilities.

Manage delivery service required server capabilities

Users with the Operations or Admin Role can associate one or more required server capabilities with a delivery service by navigating to a delivery service via Services → Delivery Services and using the context menu for the delivery services table and selecting Manage Required Server Capabilities or by navigating to Services → Delivery Services → Delivery Service → More → Manage Required Server Capabilities.

Adding a required server capability to a delivery service will ensure two things:

1. Only Edge-tier caches with the required capability can be assigned to the delivery service
2. Only Mid-tier caches with the required capability will handle requests of the delivery service (if applicable)
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.
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

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.

---

**# 79: Example DNS Query to Test a New Static DNS Entry within CDN in a Box**

```
$ 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.

2. Create a Delivery Service with Type STEERING or CLIENT_STEERING in Traffic Ops.

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
Fig. 46: Creating a STEERING Delivery Service

Fig. 47: STEERING Targets
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 Service* 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!
CHAPTER 4

Developer’s Guide

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 Compose¹

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. Suppresses 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_build²
- grovetc cfg_build
- source²
- traffic_monitor_build²
- traffic_ops_build²

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
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:
# 80: Run from the Repository’s Root Directory

```
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

At its current stage in development, “Traffic Ops” actually refers to a concept with two implementations. The original Traffic Ops was written as a collection of Perl scripts based on the Mojolicious framework framework. At some point, the relatively poor performance and lack of knowledgeable developers as the project expanded became serious issues, and so for the past few years Traffic Ops has been undergoing a steady rewrite to Go.

### Introduction

Traffic Ops at its core is mainly a PostgreSQL database used to store configuration information for ATC, and a set of RESTful API endpoints for interacting with and manipulating that information. It also serves as the single point of authentication for ATC components (that is, when one hears “user” in an ATC context it nearly always means a “user” as configured in Traffic Ops) and provides interfaces to other ATC components by proxy. Additionally, there is some miscellaneous, at times obscure functionality to Traffic Ops, such as generating arbitrary Linux system images.

### Software Requirements

Traffic Ops is only supported on CentOS 7+ systems (although many developers do use Mac OS with some success).

The two different implementations have different requirements, but they do share a few:

- **Goose** (although the `postinstall` Perl script will install this if desired)
- **PostgreSQL 9.6.6** - the machine where (either implementation of) Traffic Ops is running must have the client tool set (e.g. `psql(1)`), but the actual database can be run anywhere so long as it is accessible.
- `openssl(1SSL)` is recommended to generate server certificates, though not strictly required if certificates can be obtained by other means.
- Some kind of SMTP server is required for certain Traffic Ops API endpoints to work, but for purposes unrelated to them an SMTP server is not required.
Tip: Alternatively, development and testing can be done using CDN in a Box - albeit somewhat more slowly.

Perl Implementation Requirements

Most dependencies are managed by Carton 1.0.12, but there are some - outside of those shared with the Go implementation - that are not managed by that system.

- Carton itself
- Perl 5.10.1
- libpcap and libpcap development library - usually libpcap-dev or libpcap-devel in your system’s native package manager.
- libpq and libpq development library - usually libpq-dev or libpq-devel in your system’s native package manager.
- The JSON Perl pod from CPAN
- The JSON::PP Perl pod from CPAN
- Developers should use Perltidy to format their Perl code.

# Example Perltidy Configuration (usually in HOME/.perltidyrc)

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

- Go 1.11

- If the system’s Go compiler doesn’t provide it implicitly, also note that all Go code in the ATC repository should be formatted using `gofmt`

All Go code dependencies are managed through the `vendor/` directory and should thus be available without any extra work - and any new dependencies should be properly “vendored” into that same, top-level directory. Some dependencies have been “vendored” into `traffic_ops/vendor` and `traffic_ops/traffic_ops_golang/vendor` but the preferred location for new dependencies is under that top-level `vendor/` directory.

Per the Go language standard’s authoritative source’s recommendation, all sub-packages of `golang.org/x` are treated as a part of the compiler, and so need not ever be “vendored” as though they were an external dependency. These dependencies are not listed explicitly here, so it is strongly advised that they be fetched using `go-get(1)` rather than downloaded by hand.

Tip: All new dependencies need to be subject to community review to ensure necessity (because it will be added in its entirety to the repository, after all) and license compliance via the developer mailing list `<mailto:dev@trafficcontrol.apache.org>`.

Traffic Ops Project Tree Overview

- `traffic_ops/` - The root of the Traffic Ops project
  - `app/` - Holds most of the Perl code base, though many of the files contained herein are also used by the Go implementation

Note: This directory is home to many things that no longer work as intended or have been superseded by other things - most notably code for the now-removed Traffic Ops UI. That does not, however, mean that they are safe to remove. The API code that is still relied upon today is deeply entangled with the UI code, and in a dynamic language like Perl it can be very dangerous to remove things, because it may not be apparent that something is broken until it’s already in production. So please don’t remove anything in here until we’re ready to excise the Perl implementation entirely.

- `bin/` - Directory for scripts and tools, `cron(8)` jobs, etc.
  - `checks/` - Contains the Check Extensions scripts that are provided by default
  - `db/` - Contains scripts that manipulate the database beyond the scope of setup, migration, and seeding
tests/ - Integration and unit test scripts for automation purposes - in
general this has been superseded by traffic_ops/testing/api/

* conf/ - Aggregated configuration for Traffic Ops. For convenience,
different environments for the app/db/admin tool are already set up
  - development/ - Configuration files for the “development” environ-
ment
  - integration/ - Configuration files for the “integration” environment
  - misc/ - Miscellaneous configuration files.
  - production/ - Configuration files for the “production” environment
  - test/ - Configuration files for the “test” environment

* db/ - Database setup, seeding, and upgrade/downgrade helpers
  - migrations/ - Database migration files
  - tools/ - Contains helper scripts for easing upgrade transitions when
    selective data manipulation must be done to achieve a desirable state

* lib/ - Contains the main handling logic for the Perl implementation
  - API/ - Mojolicious Controllers for the Traffic Ops API
  - Common/ - Code that is shared between both the Traffic Ops API
    and the now-removed Traffic Ops UI
  - Connection/ - Adapter definitions for connecting to external ser-
    vices
  - Extensions/ - Contains Data Source Extensions
  - Fixtures/ - Test-case fixture data for the “testing” environment
  - Integration/ - Integration tests
  - Helpers/ - Contains route handlers for the Traffic Stats-related end-
    points
  - MojoPlugins/ - Mojolicious Plugins for common controller code
  - Schema/Result/ - Contains schema definitions generated from a
    constructed database for use with the DBIx Perl pod suite. These
    were machine-generated in 2016 and absolutely no one should be
    touching them ever again.
  - /Test - Common helpers for testing
  - UI/ - Mojolicious controllers for the now-removed Traffic Ops UI
  - Utils/ - Contains helpful utilities for certain objects and tasks

---

1 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.
· Helper/ - Common utilities for the Traffic Ops application

* public/ - A directory from which files are served statically over HTTP by the Perl implementation. One common use-case is for hosting a Coverage Zone File for Traffic Router.

* script/ - Mojolicious bootstrap/startup scripts.

* t/ - Unit tests for both the API (in api/) and the now-removed Traffic Ops UI

  · api/ - Unit tests for the API

* t_integration/ - High-level integration tests

* templates/ - Mojolicious Embedded Perl (template name.ep) files for the now-removed Traffic Ops UI

  – build/ - Contains files that are responsible for packaging Traffic Ops into an RPM file - and also for doing the same with ORT

  – client/ - The Go client library for Traffic Ops

  – etc/ - Configuration files for various systems associated with running production instances of Traffic Ops, which are installed under /etc by the Traffic Ops RPM

  · cron.d/ - Holds specifications for cron(8) jobs that need to be run periodically on Traffic Ops servers

  Note: At least one of these jobs expects itself to be run on a server that has the Perl implementation of Traffic Ops installed under /opt/traffic_ops/. Nothing terrible will happen if that’s not true, just that it/they won’t work. Installation using the RPM will set up all of these kinds of things up automatically.

  – 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

  – 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/ - Contains things that need to be accessible by the running server for certain functionality - typically installed to /var/www/data by the RPM (hence the name).

  · etc/ - This directory left empty; it’s used to contain post-installation extensions and resources
* lib/ - Contains libraries used by the various installation binaries
  – ort/ - Contains ORT and ATS configuration file-generation logic and tooling
  – testing/ - Holds utilities for testing the Traffic Ops API

* api/ - Integration testing for the Traffic Ops Go client and Traffic Ops
  – compare/ - Contains The Compare Tool

– traffic_ops_golang/ - The root of the Go implementation’s code-base

Note: The vast majority of subdirectories of traffic_ops/
traffic_ops_golang/ contain handlers for the Traffic Ops API,
and are named according to the endpoint they handle. What follows
is a list of subdirectories of interest that have a special role (i.e. don’t handle a Traffic Ops API endpoint).

See also:
The GoDoc documentation for this package

* api/ - A library for use by Traffic Ops API handlers that provides
  helpful utilities for common tasks like obtaining a database trans-
  action handle or accessing Traffic Ops configuration

* auth/ - Contains definitions of privilege levels and access control
code used in routing and provides a library for dealing with pass-
word and token-based authentication

* config/ - Defines configuration structures and methods for reading
  them in from files

* dbhelpers/ - Assorted utilities that provide functionality for com-
  mon database tasks, e.g. “Get a user by email”

* plugin/ - The Traffic Ops plugin system, with examples

* riaksvc/ - In addition to handling routes that deal with storing se-
  crets in or retrieving secrets from Traffic Vault, this package pro-
  vides a library of functions for interacting with Traffic Vault for
  other handlers to use.

* routing/ - Contains logic for mapping all of the Traffic Ops API
  endpoints to their handlers, as well as proxying requests back to
  the Perl implementation and managing plugins, and also provides
  some wrappers around registered handlers that set common HTTP
  headers and connection options

* swaggerdocs/ A currently abandoned attempt at defining the Traffic
  Ops API using Swagger - it may be picked up again at some
  point in the (distant) future
Apache Traffic Control Documentation, Release 4.0.0

* tenant/ - Contains utilities for dealing with Tenantable resources, particularly for checking for permissions

* tocookie/ - Defines the method of generating the mojolicious cookie used by Traffic Ops for authentication

* vendor/ - contains “vendored” Go packages from third party sources
  
  – vendor/ - contains “vendored” Go packages from third party sources

### 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

(Defaul: 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
### 82: Example Usage

```
db/admin --env=test reset
```

The environments are defined in the `traffic_ops/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.** In most cases it is best to clone this directly into `GOPATH/src/github.com/apache/trafficcontrol`, as otherwise the Go implementation will not function properly.

2. **Install the local dependencies using Carton.**

```
# assuming current working directory is the repository
→root
cd traffic_ops/app
carton
```

3. **Install any required Go dependencies - the suggested method is using `go-get(1)`.**
4. Set up a role (user) in PostgreSQL

See also:

PostgreSQL instructions on setting up a database.

5. Use the reset and upgrade commands of admin (see app/db/admin for usage) to set up the traffic_ops database(s).

6. Run the traffic_ops/install/bin/postinstall script, it will prompt for information like the default user login credentials

7. To run Traffic Ops, follow the instructions in Running.

Test Cases

Perl Tests

Use prove (should be installed with Perl) to execute test cases\(^1\). 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/

Go Tests

Many (but not all) endpoint handlers and utility packages in the Go code-base define Go unit tests that can be run with `go-test(1)`. There are integration tests for the Traffic Ops Go client in traffic_ops/testing/api/.

# 85: Sample Run of Go Unit Tests

```
cd traffic_ops/traffic_ops_golang

# run just one test
go test ./about

# run all of the tests
go test ./...
```

There are a few prerequisites to running the Go client integration tests:
• A PostgreSQL server must be accessible and have a Traffic Ops database schema set up (though not necessarily populated with anything).

• A running Traffic Ops Go implementation instance must be accessible - it shouldn’t be necessary to also be running the Perl implementation.

Note: For testing purposes, SSL certificates are not verified, so self-signed certificates will work fine.

Note: It is highly recommended that the Traffic Ops instance be run on the same machine as the integration tests, as otherwise network latency can cause the tests to exceed their threshold time limit of ten minutes.

The integration tests are run using `go-test(1)`, with two configuration options available.

Note: It should be noted that the integration tests will output thousands of lines of highly repetitive text not directly related to the tests its running at the time - even if the `–v` flag is not passed to `go-test(1)`. This problem is tracked by Issue #4017.

Warning: Running the tests will wipe the connected database clean, so do not ever run it on an instance of Traffic Ops that holds meaningful data.

```
--cfg CONFIG
   Specify the path to the Test Configuration File. If not specified, it will attempt to read a file named traffic-ops-test.conf in the working directory.

   See also:
   Configuring the Integration Tests for a detailed explanation of the format of this configuration file.

--fixtures FIXTURES
   Specify the path to a file containing static data for the tests to use. This should almost never be used, because many of the tests depend on the data having a certain content and structure. If not specified, it will attempt to read a file named tc-fixtures.json in the working directory.
```

### Configuring the Integration Tests

Configuration is mainly done through the configuration file passed as `--cfg`, but is also available through the following environment variables.

**SESSION_TIMEOUT_IN_SECS**
Sets the timeout of requests made to the Traffic Ops instance, in seconds.
TODB_DESCRIPTION
An utterly cosmetic variable which, if set, gives a description of the PostgreSQL database to which the tests will connect. This has no effect except possibly changing one line of debug output.

TODB_HOSTNAME
If set, will define the FQDN at which the PostgreSQL server to be used by the tests resides².

TODB_NAME
If set, will define the name of the database to which the tests will connect².

TODB_PASSWORD
If set, defines the password to use when authenticating with the PostgreSQL server.

TODB_PORT
If set, defines the port on which the PostgreSQL server listens².

TODB_SSL
If set, must be one of the following values:
- **true**  The PostgreSQL server to which the tests will connect uses SSL on the port on which it will be contacted.
- **false** The PostgreSQL server to which the tests will connect does not use SSL on the port on which it will be contacted.

TODB_TYPE
If set, tells the database driver used by the tests the kind of SQL database to which they are connecting². This author has no idea what will happen if this is set to something other than Pg, but it’s possible the tests will fail to run. Certainly never do it.

TODB_USER
If set, defines the user as whom to authenticate with the PostgreSQL server.

TO_URL
If set, will define the URL at which the Traffic Ops instance is running - including port number.

TO_USER_ADMIN
If set, will define the name of a user with the “admin” **Role** that will be created by the tests³.

TO_USER_DISALLOWED
If set, will define the name of a user with the “disallowed” **Role** that will be created by the tests³.

TO_USER_EXTENSION
If set, will define the name of a user with the “extension” **Role** that will be created by the tests³.

² The Traffic Ops instance *must* be using the same PostgreSQL database that the tests will use.
³ This does not need to match the name of any pre-existing user.
Caution: Due to legacy constraints, the only truly safe value for this is `extension` - anything else could cause the tests to fail.

**TO_USER_FEDERATION**
If set, will define the name of a user with the “federation” `Role` that will be created by the tests.

**TO_USER_OPERATIONS**
If set, will define the name of a user with the “operations” `Role` that will be created by the tests.

**TO_USER_PASSWORD**
If set, will define the password used by all users created by the tests. This does not need to be the password of any pre-existing user.

**TO_USER_PORTAL**
If set, will define the name of a user with the “portal” `Role` that will be created by the tests.

**TO_USER_READ_ONLY**
If set, will define the name of a user with the “read-only” `Role` that will be created by the tests.

**Test Configuration File**

The configuration file for the tests (defined by `--cfg`) is a JSON-encoded object with the following properties.

| Warning: Many of these configuration options are overridden by variables in the execution environment. Where this is a problem, there is an associated warning. In general, this issue is tracked by Issue #3975. |

| default | An object containing sub-objects relating to default configuration settings for connecting to external resources during testing |

| logLocations | An object containing key/value pairs where the keys are log levels and each associated value is the file location to which logs of that level will be written. The allowed values respect the reserved special names used by the `github.com/apache/trafficcontrol/lib/go-log` package. Omitted keys are treated as though their values were `null`, in which case that level is written to `/dev/null`. The allowed keys are: |

- debug
- error
- event
session An object containing key/value pairs that define the default settings used by Traffic Ops “session” connections.

**timeoutInSecs** At the time of this writing this is the only meaningful configuration option that may be present under *session*. It specifies the timeouts used by client connections during testing as an integer number of seconds. The default if not specified (or overridden) is 0, meaning no limit.

```
Warning: This configuration is overridden by SESSION_TIMEOUT_IN_SECS.
```

trafficOps An object containing information that defines the running Traffic Ops instance to use in testing.

**password** This password will be used for all created users used by the test suite - it does not need to be the password of any pre-existing user. The default if not specified (or overridden) is an empty string, which may or may not cause problems.

```
Warning: This is overridden by TO_USER_PASSWORD.
```

URL The network location of the running Traffic Ops server, including schema, hostname and optionally port number e.g. `https://localhost:6443`.

```
Warning: This is overridden by TO_URL.
```

users An object containing key-value pairs where the keys are the names of *Roles* and the values are the usernames of users that will be created with the associated *Role* for testing purposes. *There are very few good reasons why the values should not just be the same as the keys.* The default for any missing (and not overridden) key is the empty string which is won’t work so please don’t leave any undefined. The allowed keys are:

- admin
trafficOpsDB  An object containing information that defines the database to use in testing.

dbname  The name of the database to which the tests will connect.
**Warning:** This is overridden by `TODB_NAME`.

**description** An utterly cosmetic option that need not exist at all which, if set, gives a description of the database to which the tests will connect. This has no effect except possibly changing one line of debug output.

**Warning:** This is overridden by `TODB_DESCRIPTION`.

**hostname** The FQDN of the server on which the database is running.

**Warning:** This is overridden by `TODB_HOSTNAME`.

**password** The password to use when authenticating with the database

**Warning:** This is overridden by `TODB_PASSWORD`.

**port** The port on which the database listens for connections - as a string

**Warning:** This is overridden by `TODB_PORT`.

**type** The “type” of database being used. This should never be set to anything besides "Pg", anything else results in undefined behavior (although it’s equally possible that it simply won’t have any effect).

**Warning:** This is overridden by `TODB_TYPE`.

**ssl** An optional boolean value that defines whether or not the database uses SSL encryption for its connections - default if not specified (or overridden) is `false`
user The name of the user as whom to authenticate with the database

Warning: This is overridden by TODB_USER.

Writing New Endpoints

All new Traffic Ops API endpoints should be written in Go, so writing endpoints for the Perl implementation is not discussed here. Furthermore, most new endpoints are accompanied by database schema changes which necessitate a new migration under traffic_ops/app/db/migrations and database best-practices are not discussed in this section.

The first thing to consider when writing a new endpoint is what the requests it will serve will look like. It’s recommended that new endpoints avoid using “path parameters” when possible, and instead try to utilize request bodies and/or query string parameters. For example, instead of /foos/{{ID}} consider simply /foos with a supported id query parameter. The request methods should be restricted to the following, and respect each method’s associated meaning.

DELETE Removes a resource or one or more of its representations from the server. This should always be the method used when deleting objects.

GET Retrieves a representation of some resource. This should always be used for read-only operations and note that the requesting client never expects the state of the server to change as a result of a request using the GET method.

POST Requests that the server process some passed data. This is used most commonly to create new objects on the server, but can also be used more generally e.g. with a request for regenerating encryption keys. Although this isn’t strictly creating new API resources, it does change the state of the server and so this is more appropriate than GET.

PUT Places a new representation of some resource on the server. This is typically used for updating existing objects. For creating new representations/objects, use POST instead. When using PUT note that clients expect it to be idempotent, meaning that subsequent identical PUT requests should result in the same server state. What this means is that it’s standard to require that all of the information defining a resource be provided for each request even if the vast majority of it isn’t changing.

The HEAD and OPTIONS request methods have default implementations for any properly defined Traffic Ops API route, and so should almost never be defined explicitly. Other request methods (e.g. CONNECT) are currently unused and ought to stay that way for the time being.

Note: Utilizing the PATCH method is unfeasible at the time of this writing but progress toward supporting it is being made, albeit slowly in the face of other priorities.

See also:

The MDN (Mozilla Developer Network)’s documentation on the various HTTP request methods.
The final step of creating any Traffic Ops API endpoint is to write documentation for it. When doing so, be sure to follow all of the guidelines laid out in Documentation Guidelines. If documentation doesn’t exist for new functionality then it has accomplished nothing because no one using Traffic Control will know it exists. Omitted documentation is how a project winds up with a dozen different API endpoints that all do essentially the same thing.

Framework Options

The Traffic Ops code base offers two basic frameworks for defining a new endpoint. Either one may be used at the author’s discretion (or even neither if desired and appropriate - though that seems unlikely).

Generic “CRUDer”

The “Generic ‘CRUDer’”, as it’s known, is a pattern of API endpoint development that principally involves defining a type that implements the github.com/apache/trafficcontrol/traffic_ops/traffic_ops_golang/api.CRUDer interface. A description of what that entails is best left to the actual GoDoc documentation.

See also:

The github.com/apache/trafficcontrol/traffic_ops/traffic_ops_golang/api.GenericCreate, github.com/apache/trafficcontrol/traffic_ops/traffic_ops_golang/api.GenericDelete, github.com/apache/trafficcontrol/traffic_ops/traffic_ops_golang/api.GenericRead, and github.com/apache/trafficcontrol/traffic_ops/traffic_ops_golang/api.GenericUpdate helpers are often used to provide the default operations of creating, deleting, reading, and updating objects, respectively. When the API endpoint being written is only meant to perform these basic operations on an object or objects stored in the database, these should be totally sufficient.

This method offers a lot of functionality “out-of-the-box” as compared to the APIInfo method, but because of that is also restrictive. For example, it is not possible to write an endpoint that returns data not encoded as JSON using this method. That’s an uncommon use-case, but not unheard-of.

This method is best used for basic creation, reading, update, and deletion operations performed on simple objects with no structural differences across API versions.

APIInfo

Endpoint handlers can also be defined by simply implementing the net/http.HandlerFunc interface. The net/http.Request reference passed into such handlers provides identifying information for the authenticated user (where applicable) in its context.

To easily obtain the information needed to identify a user and their associated permissions, as well as server configuration information and a database transaction han-
dle, authors should use the `github.com/apache/trafficcontrol/traffic_ops/traffic_ops_golang/api.NewInfo` function which will return all of that information in a single structure as well as any errors encountered during the process and an appropriate HTTP response code in case of such errors.

This method offers fine control over the endpoint’s logic, but tends to be much more verbose than the endpoints written using the `Generic “CRUDer”` method. For example, a handler for retrieving an object from the database and returning it to the requesting client encoded as JSON can be twenty or more lines of code, whereas a single call to `github.com/apache/trafficcontrol/traffic_ops/traffic_ops_golang/api.GenericCreate` provides equivalent functionality.

This method is best used when requests are meant to have extensive side-effects, are performed on unusually structured objects, need fine control of the HTTP headers/options, or operate on objects that have different structures or meanings across API versions.

**Rewriting a Perl Endpoint**

When rewriting endpoints from Perl, some special considerations must be taken.

- Any rules and guidelines herein outlined that are broken by the Perl handler *must also be* broken in the rewritten Go handler to maintain compatibility within the API. New features can be added in the latest unreleased version of the API so long as they are appropriately documented, but avoid the temptation to fix things that seem broken. Such changes are best left to re-implementation of the API in a subsequent major version. The exceptions to this rule are if the broken behavior constitutes a security vulnerability (in which case be sure to follow the instructions on the Apache Software Foundation security page) or if it happens in the event of a server or client error. For example, many Perl handlers will spit out an HTML page in the event of a server-side error while the standard behavior of the Traffic Ops API in such cases is to return the appropriate HTTP response code and a response body containing a JSON-encoded alerts object describing the nature of the error.

- Mark newly rewritten endpoints in their `traffic_ops/traffic_ops_golang/routing/routes.go` definition with `perlBypass` to ensure that upon upgrading it is possible to configure the server to fall back on the Perl implementation. That way, any erroneous rewrites that wind up in production environments can be quickly bypassed in favor of the old, known-to-be-working version.

- The Perl handlers support any combination of optional trailing `/` and `.json` on endpoint routes, and rewritten route definitions ought to support that. For example, the endpoint `/foo` can with equal validity from the Perl implementation’s perspective as `/foo.json`, `/foo/`, `/foo.json/` (for some reason), and even (horrendously) as `/foo/./json`.

- It’s possible that a route definition for the newly rewritten route already exists, explicitly defining a proxy to the Perl implementation using `handlerToFunc(proxyHandler)` to avoid collisions with later-defined routes. These will need to be deleted in order for the route to be properly handled.
Extensions

Both the Perl and Go implementation support different kinds of extensions.

What’s typically meant by “extension” in the context of Traffic Ops is a Check Extensions which provides data for server “checks” which can be viewed in Traffic Portal under Monitor → Cache Checks. This type of extension need not know nor even care which implementation it is being used with, as it interacts with Traffic Ops through the Traffic Ops API. These are described in Legacy Perl Extensions as their description remains rather Perl-centric, but in principle their operation is not limited to the context of the Perl Implementation.

Both Perl and Go also support overrides or new definitions for non-standard Traffic Ops API routes. It is strongly recommended that no Perl-based extensions of this type be written, but for posterity they are described in Legacy Perl Extensions. The Go implementation refers to this type of “extension” as a “plugin,” and they are described in Go Plugins.

Go Plugins

A plugin is defined by a Go source file in the traffic_ops/traffic_ops_golang/plugin directory, which is expected to be named plugin name.go. A plugin is registered to Traffic Ops by a call to github.com/apache/trafficcontrol/traffic_ops/traffic_ops_golang/plugin.AddPlugin in the source file’s special init function.

A plugin is only enabled at runtime if its name is present in the cdn.conf file’s traffic_ops_golang.plugins array.

Each plugin may also define any, all, or none of the lifecycle hooks provided: load, startup, and onRequest.

load The load function of a plugin, if defined, needs to implement the github.com/apache/trafficcontrol/traffic_ops/traffic_ops_golang/plugin.LoadFunc interface, and will be run when the server starts and after configuration has been loaded. It will be passed the plugins own configuration as it was defined in the cdn.conf file’s traffic_ops_golang.plugin_config map.

onRequest The onRequest function of a plugin, if defined, needs to implement the github.com/apache/trafficcontrol/traffic_ops/traffic_ops_golang/plugin.OnRequestFunc interface, and will be called on every request made to the Traffic Ops API. Because of this, it’s imperative that the function exit as soon as possible. Note that once one plugin reports that it has served the request, no others will be tried. The order in which plugins are tried is defined by their order in the traffic_ops_golang.plugins array of the cdn.conf configuration file.

See also:

It’s very common for this function to behave like a Traffic Ops API endpoint, so when writing a plugin it may be useful to review Writing New Endpoints.

startup Like load, the startup function of a plugin, if defined, will be called when the server starts and after configuration has been loaded. Unlike load, however,
this function should implement the `github.com/apache/trafficcontrol/traffic_ops/traffic_ops_golang/plugin.StartupFunc` interface and will be passed in the entirety of the server’s configuration, including its own configuration and any shared plugin configuration data as defined in the `cdn.conf` file’s `traffic_ops_golang.plugin_shared_config` map.

Example

An example “Hello World” plugin that serves the `/_hello` request path by just writing “Hello World” in the body of a 200 OK response back to the client is provided in `traffic_ops/traffic_ops_golang/plugin/hello_world.go`:

```go
package plugin

/*
 * Licensed 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.
 */

import (  
    "strings"
)

func init() {  
    AddPlugin(10000, Funcs{onRequest: hello}, "example HTTP plugin", "1.0.0")
}

const HelloPath = "/_hello"

func hello(d OnRequestData) IsRequestHandled {  
    if !strings.HasPrefix(d.R.URL.Path, HelloPath) {  
        return RequestUnhandled
    }  
    d.W.Header().Set("Content-Type", "text/plain")
    d.W.Write([]byte("Hello, World!"))
    return RequestHandled
}

(continues on next page)
Legacy Perl 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`.

```
# 86: 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_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.

**See also:**

Contributions to traffic_ops_ort should follow the ATC Python contribution guidelines

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.

#### # 87: Legacy Call Signature

```
traffic_ops_ort [-k] [-h] [-v] [--dispersion DISP] [--login__dispersion DISP]
               [--retries RETRIES] [--wait_for_parents INT] [--rev_proxy__disable]
               [--ts_root PATH] MODE LOG_LEVEL TO_URL LOGIN```

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`, `--to_user`, and `--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.

#### # 88: 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_prox_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_or 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_or 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_or 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_or was invoked using the new call signature, which allows it to be overridden on the command line by the value of -u/--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_or was invoked using the new call signature, which allows it to be overridden on the command line by the value of -p/--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>
<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 <code>traffic_ops_ort</code> 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 <code>traffic_ops_ort</code> is running.</td>
</tr>
<tr>
<td>__HOSTNAME</td>
<td>The (short) hostname of the cache server on which <code>traffic_ops_ort</code> 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 <code>traffic_ops_ort</code> 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.

# 89: Example of Remap Override

```shell
# 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.
##OVERRIDDEN## map http://from.example.com/ http://to.example.com/ some_plugin.so

##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

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.

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.

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

```python
def 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**
```

```python
def 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 actualLogLevel
```

**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.

```python
def ORTException(
    Bases: Exception
)
    Signifies an ORT related error
```

```python
STATUS_FILE_DIR = '/opt/ort/status'
A constant that holds the absolute path to the status file directory
```

```python
def 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`

Maps Traffic Ops alert levels to logging levels

4.1. Developer’s Guide  
257
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(prmpt, default=None)

Utility function to get an interactive yes/no response to the prompt *prmpt*

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

### 4.1.4 Traffic Portal

#### Introduction

Traffic Portal is an AngularJS client served from a Node.js web server designed to consume the [Traffic Ops API](https://trafficops.apache.org). 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. Modify `traffic_portal/conf/config.js`:
   1. Valid SSL certificates and keys are needed for Traffic Portal to run. Generate these (e.g. using this SuperUser answer) and update `ssl`.
   2. Modify `api.base_url` to point to your Traffic Ops API endpoint.
   3. Modify `files.static` to be `./app/dist/public`.
   4. Modify `log.stream` to be `./server/log/access.log`.

7. Navigate to http(s)://localhost:[port|sslPort defined in `traffic_portal/conf/config.js`]

### 4.1.5 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 Http11Protocol 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
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/main/conf/dns.properties to core/src/test/conf/
   - copy core/src/main/conf/http.properties to core/src/test/conf/
   - copy core/src/main/conf/log4j.properties to core/src/test/conf/
   - copy core/src/main/conf/traffic_monitor.properties to core/src/test/conf/ and then edit the traffic_monitor.bootstrap.hosts property
   - copy core/src/main/conf/traffic_ops.properties to core/src/test/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 over HTTP at `http://localhost:3333` and over HTTPS at `https://localhost:3443`, 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.

# 90: 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)
--- 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,
    }
  </style>
</head>
<body>

(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 with the Name “api.port”, and the Config File “server.xml” on the Traffic Router’s Profile.

The API can be configured via HTTPS on port 3443 in /opt/traffic_router/conf/server.xml or by setting a Parameter named secure.api.port with configFile server.xml on the Traffic Router’s Profile. The post install script will generate self signed certificates at /opt/traffic_router/conf/, create a new Java Keystore named /opt/traffic_router/conf/keyStore.jks, and add the new certificate to the KeyStore. The password for the Java Keystore and the Keystore location are stored in /opt/traffic_router/conf/https.properties. To override the self signed certificates with new ones from a certificate authority, update the properties for the Keystore location and password at /opt/traffic_router/conf/https.properties.

The API can be configured via HTTPS on port 3443 in /opt/traffic_router/conf/server.xml or by setting a Parameter named secure.api.port with configFile server.xml on the Traffic Router’s Profile. When systemctl start traffic_router is run, it will generate self signed certificates at /opt/traffic_router/conf/, create a new Java Keystore named /opt/traffic_router/conf/keyStore.jks, and add the new certificate to the KeyStore. The password for the Java Keystore and the Keystore location are stored in /opt/traffic_router/conf/https.properties. To override the self signed certificates with new ones from a certificate authority, either replace the Java Keystore in the default location or update the properties for the new Keystore location and password at /opt/traffic_router/conf/https.properties and then restart the Traffic Router using systemctl.

Traffic Router API endpoints only respond to GET requests.

/crs/stats

General stats.

Request Structure

# 91: Request Example

GET /crs/stats HTTP/1.1
Host: trafficrouter.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*

Response Structure
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
    }
  }
} (continues on next page)
/crs/stats/ip/{{IP}}

Geolocation information for an IPv4 or IPv6 address.

Request Structure

Table 60: 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>

# 93: 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

# 94: 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**

# 95: 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 strings that are the *Names of Cache Groups* to which this Traffic Router is capable of routing client traffic

# 96: 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**
### Request Example

```
GET /crs/locations/caches HTTP/1.1
Host: trafficrouter.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
```

### Response Structure

```
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",
                    "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/locations/\{\{cachegroup\}\}/caches
```

A list of cache servers for this Cache Group only.
Request Structure

Table 61: 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>

# 99: 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

# 100: 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
    }
  ]
}
```

The resulting cache of the consistent hash using coverage zone file for a given client IP, Delivery Service, and request path.
Request Structure

Table 62: 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/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 63: 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 64: 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 65: 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>requestPath</td>
<td>yes</td>
<td>The... request path?</td>
</tr>
</tbody>
</table>

# 101: 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
# 102: Response Example

```http
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,
  "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,
      "longitude": "-88.0"
    }
  },
  "dispersion": {
    "limit": 1,
    "shuffled": true
  },
  "ip6RoutingEnabled": true,
  "responseHeaders": {},
  "requestHeaders": [],
  "regionalGeoEnabled": false,
  "geolocationProvider": "maxmindGeolocationService",
  "anonymousIpEnabled": false,
  "sslEnabled": true,
  "acceptHttp": true,
  "deepCache": "NEVER",
  "dns": false,
  "locationLimit": 0,
  "maxDnsIps": 0,
  "sslReady": true,
  "available": true
}
```
/crs/coveragezone/caches

A list of caches for a given Delivery Service and Cache Group.

Request Structure

Table 66: 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

/crs/coveragezone/cachelocation

The resulting Cache Group for a given client IP and Delivery Service.

Request Structure

Table 67: 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>
</tbody>
</table>

Response Structure

TBD

/crs/deepcoveragezone/cachelocation

The resulting Cache Group using the Deep Coverage Zone File (deep caching) for a given client IP and Delivery Service.

4.1. Developer's Guide
Request Structure

Table 68: 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>
</tbody>
</table>

Response Structure

TBD

/crs/consistenthash/patternbased/regex

The resulting path that will be used for consistent hashing when the given regex is applied to the given request path.

Request Structure

Table 69: 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>

# 103: Request Example

```
GET /crs/consistenthash/patternbased/regex?regex=%2F.*%3F%28%2F.*%3F%29.%5C.m3u8%29&requestPath=%2Ftext1234%2Fname%2Fasset.m3u8 HTTP/1.1
Host: localhost:3333
User-Agent: curl/7.54.0
Accept: */*
```

Response Structure
# 104: 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 70: 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 <em>Delivery Service</em> served by this Traffic Router</td>
</tr>
</tbody>
</table>

# 105: Request Example

```
GET /crs/consistenthash/patternbased/deliveryservice?
  deliveryServiceId=asdf&requestPath=%2Fsometext1234%2Fstream_name
  %2Fasset_name.m3u8 HTTP/1.1
Host: localhost:3333
User-Agent: curl/7.54.0
Accept: */*
```

## Response Structure

# 106: Response Example

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

(continues on next page)```
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 71: 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

### 4.1.6 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 Service 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 Service 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().

---

\(^1\) 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.
Stat Pipeline

poller `common/poller/poller.go:HttpPoller.Poll()`. Listens for configuration changes (from the Ops Configuration Manager), and starts its own, internal microthreads - 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()`. Fetches the given URL, and passes the returned data to the Handler, along with any errors.

handler `traffic_monitor/cache/cache.go:Handler.Handle()`. 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()`. 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()`. Same poller type as the Stat Poller pipeline, with a different handler object.
fetcher  common/fetcher/fetcher.go:HttpFetcher.Fetch(). Same fetcher type as the Stat Poller pipeline, with a different handler object.

handler  traffic_monitor/cache/cache.go:Handler.Handle(). 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(). 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 (lastHealthDurationsThreadsae, 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.

Aggregated Availability Data
(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

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

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

---

# 107: Example Response

```
{
  "events": [
    {
      "time": 1538417713,
      "index": 67848,
      "description": "REPORTED - loadavg too high (36.37 → 25.00) (health)",
      "name": "edge",
      "hostname": "edge",
      "type": "EDGE",
      "isAvailable": false
    }
  ]
}
```

/publish/CacheStats

Statistics gathered for each cache.

**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>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

# 108: Example Response

```json
{}
```

publish/CacheStats/{{cache}}

Statistics gathered for only a single cache.

GET

Response Type Object

Request Structure

Table 73: 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 74: 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

# 109: Example Response

```json
{}
```

/publish/DsStats

Statistics gathered for Delivery Services
GET

Response Type  Object

Request Structure

Table 75: Request Query Parameters

<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 76: 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 <em>Delivery Service</em> to inspect</td>
</tr>
</tbody>
</table>

Table 77: 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 ?
Request Structure

Table 78: 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/StatSummary

The summary of cache server statistics.

GET

Response Type 

Request Structure

Table 79: 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

TODO

4.1.7 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.
4.1.8 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
- URL
- URI
- XML
- YAML

Please do not abbreviate Traffic Control terms e.g. Cache Group, Delivery Service. See Terms for the proper way to use these terms.
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.

```
# 110: Example Usage of the code-block Directive
.. code-block:: syntax
   :caption: A short, meaningful caption
   :linenos:

   ```linenos:
   is an optional field which will include line numbers in the listing. 'syntax'
   should be the name of a valid Pygments syntax.
```

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-STRUCTURED TEXT 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>).

        # 111: Document Title

        ***************
        Document Title
        ***************

4.1. Developer’s Guide
2. Section header. This should represent a main topic of the page

# 112: Section Header

Section Header
==============

3. Subsection header. This should represent a key piece of a main topic on the page

# 113: 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.

# 114: 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.

# 115: 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”.

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. \texttt{`Cache Group`} will render as: \textit{Cache Group}. Generally speaking, be wary of using the word “cache”. To most people that means the \textit{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 \textit{Formatting}. Maintain the structural format of the API documentation as outlined in the \textit{Traffic Ops API} section. API routes that have variable paths e.g. \texttt{profiles/{{ID}}} should use mustache templates \textbf{not} the Mojolicious-specific \texttt{: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 \texttt{/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 \texttt{.. 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 \texttt{.. versionchanged} directive. If an endpoint should not be used because newer endpoints provide the same functionality in a better way, use the \texttt{.. deprecated} directive to link to them and explain why they are better.

When documenting an API route, be sure to include \textit{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 \texttt{Profile ID} in \texttt{profiles/{{ID}}}, consider that the endpoint path cannot be formed without defining \textit{all} path parameters, and so to label them as “required” is superfluous.

The “Response Example” must \textbf{always} exist. “TODO” is \textbf{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 \texttt{PATCH}, \texttt{POST} and \texttt{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 \textbf{must} be the appropriate response \textbf{to that request}. When generating Request/Response Examples, attempt to use the \textit{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 \textit{complete HTTP stack}, which captures behavior like Query Parameters, Path Parameters and HTTP cookie operations like those used by e.g. \texttt{logs}. A few caveats to the “include all headers” rule:

- The \texttt{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.

```
# 116: 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

godoc A text role that can be used to easily link to the documentation for any Go package, type, or function/method (grouped constants/variables not supported). For example, `:godoc:`net/http.HandlerFunc` renders as net/http.HandlerFunc.

atc-godoc This is provided for convenience, and is identical to :godoc: except that it is assumed to be relative to the Apache Traffic Control project. For example, `:atc-godoc:`lib/go-rfc.MimeType.Quality` renders as github.com/apache/trafficcontrol/lib/go-rfc.MimeType.Quality.

to-godoc This is provided for convenience, and is identical to :godoc: except that it is assumed to be relative to the github.com/apache/trafficcontrol/traffic_ops/traffic_ops_golang package. For example, `:to-godoc:`api.APIInfo` renders as github.com/apache/trafficcontrol/traffic_ops/traffic_ops_golang/api.APIInfo.
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 80: 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>

# 117: 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

# 118: 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=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly

(continues on next page)
POST

Deprecated since version 1.1: This endpoint does not have an alternative. API Capabilities must be seeded when new endpoints are created.

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

# 119: Request Example

```bash
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

# 120: Response Example

```bash
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)
```

(continues on next page)
apiCapabilities/{{ID}}

Manages a specific API capability.

GET

Deprecated since version ATCv4.

Get an API-capability mapping by id.

Auth. Required Yes
Roles Required None
Response Type Array

Request Structure

Table 81: 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>
# 121: Request Example

```
GET /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

- **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

# 122: 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=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: SMSFH0cD6VfJkmcHmHBQcN+jkRRCmFzx1jyBWhJeyPpO4YHPSvUcjZzs1WlJqyjwWeoXeNVwxhRkBwVg==
Content-Length: 162
```
PUT

Deprecated since version 1.1: This endpoint does not have an alternative. API Capabilities can only be modified at the database seeding level.

Edit an API-capability mapping.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** 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 82: 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>

# 123: 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

{
    "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

# 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: *
```
DELETE

Deprecated since version 1.1: This endpoint does not have an alternative. API Capabilities can only be deleted at the database seeding level.

Delete a capability.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** undefined

5.1. Traffic Ops API
Request Structure

Table 83: 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>

# 125: Request Example

```bash
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

# 126: 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, 07 Nov 2018 15:44:14 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: eTFJkB2Bh8SCT2A29e21e8qoEdNzFfuT5a3tDG7u8vwz/JHntQRRR8554ali65733uWojlWKM65bLSDNmmNqQ==
Content-Length: 73

{
    "alerts": [
        {
            "level": "success",
            "text": "API-capability mapping deleted."
        },
        {
            "level": "warning",
            "text": "This endpoint is deprecated, please use [NO ALTERNATE - See https://traffic-control-cdn.readthedocs.io/en/latest/api/api_capabilities_id.html#delete] instead"
        }
    ]
}
asns

See also:
The Autonomous System Wikipedia page for an explanation of what an ASN actually is.

GET

List all ASNs.

Auth. Required  Yes
Roles Required None
Response Type  Array

Changed in version 1.2: Previously was an object with only one key (“asns”) that contained the response array. This has been flattened so that the response is the actual 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>The ID of a Cache Group - only ASNs for this Cache Group will be returned.</td>
</tr>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

# 127: Request Example

```
GET /api/1.4/asns HTTP/1.1
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: mojolicious=...
```
Response Structure

- **asn**: An ASN as specified by IANA for identifying a service provider.
- **cachegroup**: A string that is the Name of the Cache Group that is associated with this ASN.
- **cachegroupId**: An integer that is the ID of the Cache Group that is associated with this ASN.
- **id**: An integral, unique identifier for this association between an ASN and a Cache Group.
- **lastUpdated**: The time and date this server entry was last updated in an ISO-like format.

# 128: 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-Encoding: gzip
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 02 Dec 2019 22:51:14 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: F2NmDbTpXqrIQDX7IBKH9+1drtTL4XedSfJv6k1MgLEZwbLCkddIXuSLpmgVCID6kTVgy3fTkjZS3U
X-Server-Name: traffic_ops_golang/
Date: Mon, 02 Dec 2019 21:51:14 GMT
Content-Length: 128

{
    "response": [
        {
            "asn": 1,
            "cachegroup": "TRAFFIC_ANALYTICS",
            "cachegroupId": 1,
            "id": 1,
            "lastUpdated": "2019-12-02 21:49:08+00"
        }
    ]
}
```

**POST**

Creates a new ASN.

**Auth. Required**: Yes

**Roles Required**: “admin” or “operations”
Response Type  Object

Request Structure

asn  The value of the new ASN

cachegroup  An optional field which, if present, is a string that 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 is set entirely by the cachegroupId field, and so the actual Name may differ from what was in the request.

cachegroupId  An integer that is the ID of a Cache Group to which this ASN will be assigned

# 129: Request Example

```
POST /api/1.4/asns HTTP/1.1
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: mojolicious=...
Content-Length: 29

{"asn": 1, "cachegroupId": 1}
```

Response Structure

asn  An ASN as specified by IANA for identifying a service provider

cachegroup  A string that is the Name of the Cache Group that is associated with this ASN

cachegroupId  An integer that is the ID of the Cache Group that is associated with this ASN

id  An integral, unique identifier for this association between an ASN and a Cache Group

lastUpdated  The time and date this server entry was last updated in an ISO-like format
# 130: 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-Encoding: gzip
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 02 Dec 2019 22:49:08 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: mx8b2GTYojz4QtMxXCMoQyZogCB504vs0yv6WGly4dwM81W3XiejWNuUwchRBYYi8QHaWsMZ3DaigP
X-Server-Name: traffic_ops_golang/
Date: Mon, 02 Dec 2019 21:49:08 GMT
Content-Length: 150

{
  "alerts": [
    {
      "text": "asn was created.",
      "level": "success"
    }
  ],
  "response": {
    "asn": 1,
    "cachegroup": null,
    "cachegroupId": 1,
    "id": 1,
    "lastUpdated": "2019-12-02 21:49:08+00"
  }
}
```

```
asns/{{id}}
```

See also:
The Autonomous System Wikipedia page for an explanation of what an ASN actually is.

**GET**

Retrieve information about a specific ASN-to-[Cache Group] association.

- **Auth. Required**  Yes
- **Roles Required**  None
- **Response Type**  Array
Request Structure

Table 85: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
</table>
| id       | yes      | The integral, unique identifier of the desired ASN-to-
|          |          | Cache Group association                             |

# 131: Request Structure

GET /api/1.1/asns/1 HTTP/1.1
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: mojolicious=...

Response Structure

- **asn** An ASN as specified by IANA for identifying a service provider
- **cachegroup** A string that is the Name of the Cache Group that is associated with this ASN
- **cachegroupId** An integer that is the ID of the Cache Group that is associated with this ASN
- **id** An integral, unique identifier for this association between an ASN and a Cache Group
- **lastUpdated** The time and date this server entry was last updated in an ISO-like format

# 132: 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-Encoding: gzip
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 02 Dec 2019 22:53:17 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512:
X-Server-Name: traffic_ops_golang/

(continues on next page)
PUT

Allows user to edit an existing ASN-to-Cache Group association.

**Auth. Required**  Yes

**Roles Required**  “admin” or “operations”

**Response Type**  Object

Request Structure

- **asn**  The new ASN which will be associated with the identified Cache Group - must not conflict with existing associations

- **cachegroup**  An optional field which, if present, is a string that 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 is set entirely by the cachegroupId field, and so the actual Name may differ from what was in the request.

- **cachegroupId**  An integer that is the ID of a Cache Group to which this ASN will be assigned  - must not conflict with existing associations

<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-to-Cache Group association</td>
</tr>
</tbody>
</table>
# 133: Request Example

```
PUT /api/1.4/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**  An ASN as specified by IANA for identifying a service provider

- **cacheGroup**  A string that is the Name of the Cache Group that is associated with this ASN

- **cachegroupId**  An integer that is the ID of the Cache Group that is associated with this ASN

- **id**  An integral, unique identifier for this association between an ASN and a Cache Group

- **lastUpdated**  The time and date this server entry was last updated in an ISO-like format

# 134: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: 83P4LJVsrgQxKBKHFxo5pbnhQM1B4o3a9v3PkspyOJcpNx1S/GJhCPpiANBki547sY+0vTqT6IrHZ4GYp8bA==
X-Server-Name: traffic_ops_golang/
Date: Thu, 08 Nov 2018 14:37:39 GMT
Content-Length: 160

{ "alerts": [
    {
      "text": "asn was updated."
    }
]
```
"level": "success"
}
],
"response": {
  "asn": 2,
  "cachegroup": null,
  "cachegroupId": 1,
  "id": 1,
  "lastUpdated": "2018-11-08 14:37:39+00"
}}

DELETE

Deletes an association between an ASN and a Cache Group.

Auth. Required Yes
Roles Required “admin” or “operations”
Response Type undefined

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>The integral, unique identifier of the desired ASN-to-Cache Group association</td>
</tr>
</tbody>
</table>

# 135: Request Example

```
DELETE /api/1.4/asns/1 HTTP/1.1
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: mojolicious=...
Content-Length: 0
```

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, Set-Cookie, Cookie  
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE  
Access-Control-Allow-Origin: *  
Content-Encoding: gzip  
Content-Type: application/json  
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 02 Dec 2019 23:06:24 GMT; Max-Age=3600; HttpOnly  
Whole-Content-Sha512: 6t3WA+DOcfPJB5UnvDpzEVx5ySfmJgEV9wgkO71U5k32L1VXpxcaTdDVNGgDD19sdNftmYnKXf5.jpeg  
X-Server-Name: traffic_ops_golang/  
Date: Mon, 02 Dec 2019 22:06:24 GMT  
Content-Length: 81

```json
{
    "alerts": [
        {
            "text": "asn was deleted."
        },
        {
            "level": "success"
        }
    ]
}
```

cache_stats

Retrieves detailed, aggregated statistics for caches in a specific CDN.

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 88: 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>endDate</td>
<td>yes</td>
<td>The date and time until which statistics shall be aggregated in RFC 3339 format (with or without sub-second precision), the number of nanoseconds since the Unix Epoch, or in the same, proprietary format as the lastUpdated fields prevalent throughout the Traffic Ops API</td>
</tr>
<tr>
<td>exclude</td>
<td>no</td>
<td>Either “series” to omit the data series from the result, or “summary” to omit the summary data from the result - directly corresponds to fields in the Response Structure</td>
</tr>
<tr>
<td>interval</td>
<td>no</td>
<td>Specifies the interval within which data will be “bucketed”; e.g. when requesting data from 2019-07-25T00:00:00Z to 2019-07-25T23:59:59Z with an interval of “1m”, the resulting data series (assuming it is not excluded) should contain (24\text{hours} \times 60\text{minutes} \times 1\text{day} \times 1\text{minute per datapoint} = 1440\text{data points}) The allowed values for this parameter are valid InfluxQL duration literal strings matching ^d+[mhdw]$</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>A natural number indicating the maximum amount of data points should be returned in the series object</td>
</tr>
<tr>
<td>metricType</td>
<td>yes</td>
<td>The metric type being reported - one of: ‘connections’, ‘bandwidth’, ‘maxkbps’</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>A natural number of data points to drop from the beginning of the returned data set</td>
</tr>
<tr>
<td>orderBy</td>
<td>no</td>
<td>Though one struggles to imagine why, this can be used to specify “time” to sort data points by their “time” (which is the default behavior)</td>
</tr>
<tr>
<td>startDate</td>
<td>yes</td>
<td>The date and time from which statistics shall be aggregated in RFC 3339 format (with or without sub-second precision), the number of nanoseconds since the Unix Epoch, or in the same, proprietary format as the lastUpdated fields prevalent throughout the Traffic Ops API</td>
</tr>
</tbody>
</table>

---

[^1]: In this context, the number of nanoseconds per minute is calculated as follows: \(1\text{minute} = 60\text{seconds} = 60^6\text{nanoseconds}\). Therefore, the total number of nanoseconds for a minute is \(60^6\). Since there are 1440 minutes in a day, the total number of nanoseconds in a day is \(60^6 \times 1440\). Thus, for a day, the result is \(86400^6\text{nanoseconds}\).  

[^2]: The expression \(^d+[mhdw]\) indicates valid InfluxQL duration literal strings that match day, hour, minute, or second.
# 137: Request Example

```text
GET /api/1.4/cache_stats?cdnName=CDN&endDate=2019-10-28T20:49:00Z&metricType=bandwidth&startDate=2019-10-28T20:45:00Z HTTP/1.1
User-Agent: python-requests/2.20.1
Accept-Encoding: gzip, deflate
Accept: application/json;timestamp=unix, application/json;
timestamp=rfc;q=0.9, application/json;q=0.8, */*;q=0.7
Connection: keep-alive
Cookie: mojolicious=...
```

## Content Format

It’s important to note in Request Example the use of a complex “Accept” header. This endpoint accepts two special media types in the “Accept” header that instruct it on how to format the timestamps associated with the returned data. Specifically, Traffic Ops will recognize the special, optional, non-standard parameter of `application/json; timestamp`. The values of this parameter are restricted to one of

- **rfc** Returned timestamps will be formatted according to **RFC 3339** (no sub-second precision).
- **unix** Returned timestamps will be formatted as the number of nanoseconds since the Unix Epoch (midnight on January 1\(^{st}\) 1970 UTC).

## Implementation Detail

The endpoint passes back nanoseconds, specifically, because that is the form used both by InfluxDB, which is used to store the data being served, and Go’s standard library. Clients may need to convert the value to match their own standard libraries - e.g. the `Date()` class in Javascript expects milliseconds.

The default behavior - when only e.g. `application/json` or `*/*` is given - is to use **RFC 3339** formatting. It will, however, respect quality parameters. It is suggested that clients request timestamps they can handle specifically, rather than relying on this default behavior, as it is **subject to change** and is in fact **expected to invert in the next major release** as string-based time formats become deprecated.

See also:

For more information on the “Accept” HTTP header, consult its dedicated page on MDN.

## Response Structure

- **series** An object containing the actual data series and information necessary for working with it.
columns This is an array of names of the columns of the
data contained in the “values” array - should always be
["time", "sum_count"]

count The number of data points contained in the “values” array

name The name of the data set. Should always match
metric.ds.1min where metric is the requested

metricType

values The actual array of data points. Each represents a length
of time specified by the interval query parameter

time The time at which the measurement was
taken. This corresponds to the beginning
of the interval. This time comes in the
format of either an RFC 3339-formatted
string, or a number containing the num-
er of nanoseconds since the Unix Epoch
depending on the “Accept” header sent by
the client, according to the rules outlined
in Content Format.

value The value of the requested
metricType at the time given by
time. This will always be a floating point
number, unless no data is available for
the data interval, in which case it will be
null

summary A summary of the data contained in the “series” object

average The arithmetic mean of the data’s values

count The number of measurements taken within the requested
timespan. This is, in general, not the same as the count
field of the series object, as it reflects the number of
underlying, un-”bucketed” data points, and is therefore
dependent on the implementation of Traffic Stats.

fifthPercentile Data points with values less than or equal to this
number constitute the “bottom” 5% of the data set

max The maximum value that can be found in the requested
data set

min The minimum value that can be found in the requested data
set

ninetyEighthPercentile Data points with values greater than or
equal to this number constitute the “top” 2% of the data set

ninetyFifthPercentile Data points with values greater than or
equal to this number constitute the “top” 5% of the data set
# 138: 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-Encoding: gzip
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: p4asf1n7fXGtgpW/dWgo1JWdxjwDccjyvjOPFqckbgoXGUHej5/wlz7br1Q48t3ZnOWCq0lbsu2eSiBssBtUQ==
X-Server-Name: traffic_ops_golang/
Date: Mon, 28 Oct 2019 20:49:51 GMT

{
    "response": {
        "series": {
            "columns": [
                "time",
                "sum_count"
            ],
            "count": 4,
            "name": "bandwidth.cdn.1min",
            "tags": {
                "cdn": "CDN-in-a-Box"
            },
            "values": [
                [1572295500000000000, null],
                [1572295560000000000, 113.66666666666666],
                [1572295620000000000, 108.83333333333334],
                [1572295680000000000, 113]
            ]
        },
        "summary": {
```

(continues on next page)
"average": 111.83333333333333,
"count": 3,
"fifthPercentile": 0,
"max": 113.66666666666666,
"min": 108.83333333333334,
"ninetyEighthPercentile": 113.66666666666666,
"ninetyFifthPercentile": 113.66666666666666
}

```cachegroup_fallbacks
```

Deprecated since version 1.4: The cachegroups and cachegroups/{{ID}} endpoints now contain a list of Fallbacks in the output, and support it in input, and so this endpoint is redundant.

**GET**

Retrieve the Fallbacks of 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 ID of a Cache Group whose Fallbacks shall be retrieved</td>
</tr>
<tr>
<td>fallbackId</td>
<td>no</td>
<td>The integral, unique identifier of a single “fallback” Cache Group</td>
</tr>
</tbody>
</table>

**# 139: Request Example**

```
GET /api/1.4/cachegroup_fallbacks?cacheGroupId=7 HTTP/1.1
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: mojolicious=...
```
Response Structure

- **cacheGroupId** An integer that is the *ID* of the *Cache Group* described by this entry
- **cacheGroupName** The *Name* of the *Cache Group* described by this entry as a string
- **fallbackId** An integer that is the *ID* 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” as a string
- **fallbackOrder** The place in the list of *Fallbacks* of the *Cache Group* identified by `cacheGroupId` and `cacheGroupName` where the *Cache Group* identified by `fallbackId` and `fallbackName` starting from index 1.

# 140: 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-Encoding: gzip
Content-Length: 189
Content-Type: application/json
Date: Mon, 02 Dec 2019 22:26:27 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Tue, 03 Dec 2019 02:26:27 GMT; _
            path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: zSAeB8nxonyinsgl/at/10/
                     9FRRPw7N27DpkcZxRIwEzDOEY5XVfYcCHHFg1d/Q2JWtWZ9iRhs8mK5rLbKkccw==

{
  "alerts": [
    {
      "level": "warning",
      "text": "This endpoint is deprecated, please use
            'GET /cachegroups' instead"
    }
  ],
  "response": [
    {
      "cacheGroupId": 7,
      "fallbackOrder": 2,
      "fallbackName": "test",
      "fallbackId": 8,
    }
  ]
}
```

(continues on next page)
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** An integer that is the ID of a Cache Group to which to assign a fallback
- **fallbackId** An integer that is the ID of a Cache Group on which the Cache Group identified by cacheGroupId will “fall back”
- **fallbackOrder** The place in the list of Fallbacks of the Cache Group identified by cacheGroupId and cacheGroupName where the Cache Group identified by fallbackId and fallbackName starting from index 1.

# 141: Request Example

```plaintext
POST /api/1.4/cachegroup_fallbacks HTTP/1.1
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: mojolicious=...
Content-Length: 57

[["cacheGroupId": 7, "fallbackId": 8, "fallbackOrder": 2]]
```

Response Structure

- **cacheGroupId** An integer that is the ID of the Cache Group described by this entry
- **cacheGroupName** The Name of the Cache Group described by this entry as a string
**fallbackId** An integer that is the ID 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” as a string

**fallbackOrder** The place in the list of *Fallbacks* of the *Cache Group* identified by cacheGroupId and cacheGroupName where the *Cache Group* identified by fallbackId and fallbackName starting from index 1.

### # 142: 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-Encoding: gzip
Content-Length: 174
Content-Type: application/json
Date: Mon, 02 Dec 2019 22:23:22 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Tue, 03 Dec 2019 02:23:22 GMT;
path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: S8CMeR3P22itBNYOQaIjiQPMDoq2AzGt0/oBYpMPlb8/iKeZfGSS4zyt4WYbVJgrzF2YGUnhBEJe6uimQYdCQ==

{ "alerts": [
    {
      "level": "success",
      "text": "Backup configuration CREATE for cache group 7 successful."
    },
    {
      "level": "warning",
      "text": "This endpoint is deprecated, please use 'POST /cachegroups with a non-empty 'fallbacks' array' instead"
    }
]}
```

**PUT**

Updates an existing *fallback* configuration for one or more *Cache Groups*.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Array

5.1. Traffic Ops API 333
Apache Traffic Control Documentation, Release 4.0.0

Request Structure
The request payload for this endpoint must be an array, even if only one fallback relationship
is being updated.
cacheGroupId An integer that is the ID of a Cache Group to which to assign a
fallback
fallbackId An integer that is the ID of a Cache Group on which the Cache Group
identified by cacheGroupId will “fall back”
fallbackOrder The place in the list of Fallbacks of the Cache Group identified by
cacheGroupId and cacheGroupName where the Cache Group identified by fallbackId and fallbackName starting from index 1.
# 143: Request Example
PUT /api/1.4/cachegroup_fallbacks HTTP/1.1
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: mojolicious=...
Content-Length: 58
[{"cacheGroupId": 7, "fallbackId": 8, "fallbackOrder": 2}]

Response Structure
cacheGroupId An integer that is the ID of the Cache Group described by this
entry
cacheGroupName The Name of the Cache Group described by this entry as a
string
fallbackId An integer that is the ID 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” as a string
fallbackOrder The place in the list of Fallbacks of the Cache Group identified by
cacheGroupId and cacheGroupName where the Cache Group identified by fallbackId and fallbackName starting from index 1.
# 144: Response Example
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)

334

Chapter 5. APIs


DELETE

Remove one or more *Fallbacks* from one or more *Cache Groups*.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** undefined
### Request Structure

Table 90: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cacheGroupId</td>
<td>yes²</td>
<td>The ID of a Cache Group from which Fallbacks are being removed</td>
</tr>
<tr>
<td>fallbackId</td>
<td>yes²</td>
<td>The ID of a “fallback” Cache Group</td>
</tr>
</tbody>
</table>

# 145: Request Example

```
DELETE /api/1.4/cachegroup_fallbacks?fallbackId=8 HTTP/1.1
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: mojolicious=...
Content-Length: 0
```

### Response Structure

# 146: 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-Encoding: gzip
Content-Length: 186
Content-Type: application/json
Date: Mon, 02 Dec 2019 22:30:58 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Tue, 03 Dec 2019 02:30:58 GMT;
           path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: iag1k8Ym4K6nrpahJwzyA45m2RO6l59gSRg4ozUvg69/
                       TKrTLyygMeAVdzbwn8+ayOFq011TK1Ho9jQFJ5j2w==

{ "alerts": [ (continues on next page) }
```

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.
cachegroup/{{parameter ID}}/parameter

 Deprecated since version ATCv4.

**Danger:** This endpoint does not appear to work, and thus its use is strongly discouraged!

**GET**

Extract identifying information about all Cache Groups with a specific Parameter

**Auth. Required** Yes

**Roles Required** None

**Response Type** Object

**Request Structure**

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter_ID</td>
<td>yes</td>
<td>A ID</td>
</tr>
</tbody>
</table>

# 147: Request Example

```
GET /api/1.4/cachegroup/1/parameter HTTP/1.1
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: mojolicious=...
```
Response Structure

**cachegroups** An array of all *Cache Groups* with an associated *Parameter* identifiable by the `parameter_id` request path parameter

**id** An integer that is the *Cache Group’s ID*

**name** A string that is the *Name* of the *Cache Group*

# 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-Encoding: gzip
Content-Length: 161
Content-Type: application/json
Date: Tue, 03 Dec 2019 15:15:26 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Tue, 03 Dec 2019 19:15:26 GMT;
path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: H03AKuJ2IjG3wb6SEplNtIm8ka3JJdRxc2Hy0knZjHdsh8p7UcJ1teYvYUf8yMNDt8HHBaKzIDoHOh

{
    "alerts": [
        {
            "level": "warning",
            "text": "This endpoint is deprecated, please use 'GET /cachegroupparameters & GET /cachegroups' instead"
        }
    ],
    "response": {
        "cachegroups": [
            {
                "name": "CDN_in_a_Box_Edge",
                "id": 7
            },
            {
                "name": "CDN_in_a_Box_Mid",
                "id": 6
            },
            {
                "name": "TRAFFIC_ANALYTICS",
                "id": 1
            }
        ]
    }
}
```

(continues on next page)
cachegroupparameters

GET

Extract information about the Parameters associated with Cache Groups.

Auth. Required  Yes
Roles Required  None
Response Type  Object
Response Structure

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cachegroup</td>
<td>no</td>
<td>Show only the Parameters with the Cache Group identified by this integral, unique identifier</td>
</tr>
<tr>
<td>parameter</td>
<td>no</td>
<td>Show only the Parameters with the Parameter identified by this integral, unique identifier</td>
</tr>
<tr>
<td>orderby</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the nth page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

Response Structure

cachegroupParameters An array of identifying information for the Parameters of Cache Groups

cachegroup A string containing the Name of the Cache Group

last_updated Date and time of last modification in an ISO-like format

parameter An integer that is the Parameter’s ID

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 18:24:12 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly

(continues on next page)
POST

Assign Parameter(s) to Cache Group(s). Parameters already assigned to one or more Profiles cannot be assigned to Cache Groups.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Array

**Request Structure**

This endpoint accepts two formats for the request payload:

**Single Object Format** For assigning a single Parameter to a single Cache Group

**Array Format** For making multiple assignments of Parameters to Cache Groups simultaneously

### Single Object Format

**cacheGroupId** An integer that is the ID of the Cache Group to which a Parameter is being assigned

**parameterId** An integer that is the ID of the Parameter being assigned

# 150: Request Example

```
POST /api/1.1/cachegroupparameters HTTP/1.1
Host: trafficops.infra.ciab.test
```
Array Format

**cacheGroupId**  An integer that is the *ID* of the *Cache Group* to which a *Parameter* is being assigned

**parameterId**  An integer that is the *ID* of the *Parameter* being assigned

# 151: Request Example

```
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
},
{
   "cacheGroupId": 8,
   "parameterId": 125
}]
```

Response Structure

**cacheGroupId**  An integer that is the *ID* of the *Cache Group* to which a *Parameter* has been assigned

**parameterId**  An integer that is the *ID* of the *Parameter* which has been assigned
# 152: 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 15:47:49 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: wCv388wFaSjgLdCI9dchlyGxaVr81hBAGZ5F+rpI2/azCswETcY06NxsXsQrzGkluMv67J16rV+vNsQ==
Content-Length: 136

{ "alerts": [ {
    "level": "success",
    "text": "Cache group parameter associations were created."
  }
],
"response": [ {
    "cacheGroupId": 8,
    "parameterId": 124
  }
]}
```

cachegroupparameters/{{ID}}/{{parameterID}}

DELETE

Dissociate a Parameter with a Cache Group

- **Auth. Required** Yes
- **Roles Required** “admin” or “operations”
- **Response Type** undefined
Request Structure

Table 93: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The ID of the Cache Group which will have the Parameter association deleted.</td>
</tr>
<tr>
<td>parameterID</td>
<td>The ID of the Parameter which will be removed from a Cache Group</td>
</tr>
</tbody>
</table>

# 153: 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

# 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 18:26:40 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: Cuj+ZPAksDLP4FpbJDcwsWY0yVQA1Um1CWraeTIQEMlyJSBEml7oKQWDjzTrvqqV8Pzhu3g2lCoV
Content-Length: 84

{ "alerts": [ { "level": "success", "text": "cachegroup parameter was deleted." } ] }
```
GET

Extract information about Cache Groups.

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>type</td>
<td>no</td>
<td>Return only Cache Groups that are of the Type identified by this integral, unique identifier</td>
</tr>
<tr>
<td>orderBy</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

# 155: 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

f fallbacks An array of strings that are Cache Group names that are registered as Fallbacks for this Cache Group\(^1\)

\(^1\) Traffic Router will first check for a fallbacks array and, when that is empty/unset/all the Cache Groups in it are also unavailable, will subsequently check for fallbackToClosest. If that is true, then it falls back to the geographically closest Cache Group capable of serving the same content or, when it is false/no such Cache Group exists/said Cache Group is also unavailable, will respond to clients with a failure response indicating the problem.
New in version ATCv4.0: This field was added to all versions of this end-
point with ATC version 4.0

**fallbackToClosest** A boolean value that defines the *Fallback to Closest* behavior of this *Cache Group*

**id** An integer that is the *ID* of the *Cache Group*

**lastUpdated** The time and date at which this entry was last updated in an ISO-like format

**latitude** A floating-point *Latitude* for the *Cache Group*

**localizationMethods** An array of *Localization Methods* as strings

New in version ATCv4.0: This field was added to all versions of this end-
point with ATC version 4.0

**longitude** A floating-point *Longitude* for the *Cache Group*

**name** A string containing the *Name* of the *Cache Group*

**parentCachegroupId** An integer that is the *ID* of this *Cache Group’s Parent* - or null if it doesn’t have a *Parent*

**parentCachegroupName** A string containing the *Name* of this *Cache Group’s Parent* - or null if it doesn’t have a *Parent*

**secondaryParentCachegroupId** An integer that is the *ID* of this *Cache Group’s Secondary Parent* - or null if it doesn’t have a *Secondary Parent*

**secondaryParentCachegroupName** A string containing the *Name* of this *Cache Group’s Secondary Parent Cache Group* - or null if it doesn’t have a *Secondary Parent*

**shortName** A string containing the *Short Name* of the *Cache Group*

**typeId** An integral, unique identifier for the ‘*Type*’ of the *Cache Group*

**typeName** A string that names 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’.

---

# 156: Response Example

```text
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=Mon, 18 Nov 2019
  17:40:54 GMT; Max-Age=3600; HttpOnly
```

(continues on next page)
POST

Creates a Cache Group

Auth. Required  Yes
Roles Required  “admin” or “operations”
Response Type  Object

Request Structure

**fallbacks**  An optional field which, when present, should contain an array of strings that are the Names of other Cache Groups which will be the Fallbacks¹

New in version ATCv4.0: Support for this field was added to all versions of this endpoint with Apache Traffic Control version 4.0

**fallbackToClosest**  A boolean that sets the Fallback to Closest behavior of the Cache Group¹

---

Note:  The default value of fallbackToClosest is true, and if it
is null Traffic Control components will still interpret it as though it were true.

**latitude**  An optional field which, if present, should be a floating-point number that will define the *Latitude* for the *Cache Group*.

**localizationMethods**  Array of *Localization Methods* (as strings)

New in version ATCv4.0: Support for this field was added to all versions of this endpoint with ATC version 4.0.

**Tip:** This field has no defined meaning if the *Type* identified by *typeId* is not “EDGE_LOC”.

**longitude**  An optional field which, if present, should be a floating-point number that will define the *Longitude* for the *Cache Group*.

**name**  The *Name* of the *Cache Group*

**parentCachegroupId**  An optional field which, if present, should be an integer that is the *ID* of a *Parent* for this *Cache Group*.

**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 *Cache Group’s Type*

**Note:** The actual, integral, unique identifiers for these *Types* must first be obtained, generally via *types*.

---

# 157: Request Example

```plaintext
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/json

{
    "name": "test",
    "shortName": "test",
    "latitude": 0,
    "longitude": 0,
}
```

(continues on next page)

---

2 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*. 

---
Response Structure

**fallbacks**  An array of strings that are *Cache Group names* that are registered as *Fallbacks* for this *Cache Group*

New in version ATCv4.0: This field was added to all versions of this endpoint with ATC version 4.0

**fallbackToClosest**  A boolean value that defines the *Fallback to Closest* behavior of this *Cache Group*

**id**  An integer that is the *ID* of the *Cache Group*

**lastUpdated**  The time and date at which this entry was last updated in an ISO-like format

**latitude**  A floating-point *Latitude* for the *Cache Group*

**localizationMethods**  An array of *Localization Methods* as strings

New in version ATCv4.0: This field was added to all versions of this endpoint with ATC version 4.0

**longitude**  A floating-point *Longitude* for the *Cache Group*

**name**  A string containing the *Name* of the *Cache Group*

**parentCachegroupId**  An integer that is the *ID* of this *Cache Group’s Parent* - or *null* if it doesn’t have a *Parent*

**parentCachegroupName**  A string containing the *Name* of this *Cache Group’s Parent* - or *null* if it doesn’t have a *Parent*

**secondaryParentCachegroupId**  An integer that is the *ID* of this *Cache Group’s Secondary Parent* - or *null* if it doesn’t have a *Secondary Parent*

**secondaryParentCachegroupName**  A string containing the *Name* of this *Cache Group’s Secondary Parent Cache Group* - or *null* if it doesn’t have a *Secondary Parent*

**shortName**  A string containing the *Short Name* of the *Cache Group*

**typeId**  An integral, unique identifier for the ‘Type’ of the *Cache Group*

**typeName**  A string that names the *Type* of this *Cache Group*
# 158: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: YvZlh3rpfl3nBq6SbNVhbkt3IvckbB9amqGW2JhLxWK9K3cxjBq5J2sIHBUhrLKUhE9afpxtvaYrLR
X-Server-Name: traffic_ops_golang/
Date: Wed, 07 Nov 2018 22:11:50 GMT
Content-Length: 379

{
  "alerts": [
    {
      "text": "cachegroup was created.",
      "level": "success"
    }
  ],
  "response": {
    "id": 8,
    "name": "test",
    "shortName": "test",
    "latitude": 0,
    "longitude": 0,
    "parentCachegroupName": null,
    "parentCachegroupId": null,
    "secondaryParentCachegroupName": null,
    "secondaryParentCachegroupId": null,
    "fallbackToClosest": true,
    "localizationMethods": [
      "DEEP_CZ",
      "CZ",
      "GEO"
    ],
    "typeName": "EDGE_LOC",
    "typeId": 23,
    "lastUpdated": "2019-12-02 22:21:08+00",
    "fallbacks": []
  }
}
```

cachegroups/{{ID}}

Extracts information about a single Cache Group
GET

Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

Table 95: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>orderBy</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

Table 96: Request Path Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The ID of a Cache Group</td>
</tr>
</tbody>
</table>

Response Structure

**f fallbacks** An array of strings that are Cache Group names that are registered as Fallbacks for this Cache Group\(^1\)

New in version ATCv4.0: This field was added to all versions of this endpoint with ATC version 4.0

**fallbackToClosest** A boolean value that defines the Fallback to Closest behavior of this Cache Group\(^1\)

**id** An integer that is the ID of the Cache Group

\(^1\) Traffic Router will first check for a fallbacks array and, when that is empty/unset/all the Cache Groups in it are also unavailable, will subsequently check for fallbackToClosest. If that is true, then it falls back to the geographically closest Cache Group capable of serving the same content or, when it is false/no such Cache Group exists/said Cache Group is also unavailable, will respond to clients with a failure response indicating the problem.
lastUpdated The time and date at which this entry was last updated in an ISO-like format

latitude A floating-point Latitude for the Cache Group

localizationMethods An array of Localization Methods as strings

New in version ATCv4.0: This field was added to all versions of this end-point with ATC version 4.0

longitude A floating-point Longitude for the Cache Group

name A string containing the Name of the Cache Group

parentCachegroupId An integer that is the ID of this Cache Group’s Parent - or null if it doesn’t have a Parent

parentCachegroupName A string containing the Name of this Cache Group’s Parent - or null if it doesn’t have a Parent

secondaryParentCachegroupId An integer that is the ID of this Cache Group’s Secondary Parent - or null if it doesn’t have a Secondary Parent

secondaryParentCachegroupName A string containing the Name of this Cache Group’s Secondary Parent Cache Group - or null if it doesn’t have a Secondary Parent

shortName A string containing the Short Name of the Cache Group

typeId An integral, unique identifier for the ‘Type’ of the Cache Group

typeName A string that names 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’.

# 159: 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=/; Expires=Mon, 18 Nov 2019
  17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512:
  EXO+TK1CIwQ5lzTXQGqlLDzU641pLLCQbyqz528QUYSPAjjn5cqC9W3c0ioDiCdK9bUwHP3E4/
  ERBzkBTi06g==
X-Server-Name: traffic_ops_golang/
Date: Wed, 14 Nov 2018 18:35:53 GMT
Content-Length: 357
```

(continues on next page)


```json
{
  "response": [
    {
      "id": 8,
      "name": "test",
      "shortName": "test",
      "latitude": 0,
      "longitude": 0,
      "parentCachegroupName": "CDN_in_a_Box_Mid",
      "parentCachegroupId": 6,
      "secondaryParentcachegroupName": null,
      "secondaryParentCachegroupId": null,
      "fallbackToClosest": true,
      "localizationMethods": [
        "DEEP_CZ",
        "CZ"
      ],
      "typeName": "EDGE_LOC",
      "typeId": 23,
      "lastUpdated": "2018-11-14 18:23:33+00",
      "fallbacks": []
    }
  ]
}
```

**PUT**

Update *Cache Group*

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Object

**Request Structure**

Table 97: Request Path Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The ID of a <em>Cache Group</em></td>
</tr>
</tbody>
</table>

**fallbacks** An optional field which, when present, should contain an array of strings that are the *Names* of other *Cache Groups* which will be the *Fallbacks*.

New in version ATCv4.0: Support for this field was added to all versions of this endpoint with ATC version 4.0.

**fallbackToClosest** A boolean that sets the *Fallback to Closest* behavior of the *Cache Group*.

5.1. Traffic Ops API
Note: The default value of fallbackToClosest is true, and if it is null Traffic Control components will still interpret it as though it were true.

**latitude** An optional field which, if present, should be a floating-point number that will define the Latitude for the Cache Group.

**localizationMethods** Array of Localization Methods (as strings)

New in version ATCv4.0: Support for this field was added to all versions of this endpoint with ATC version 4.0.

Tip: This field has no defined meaning if the Type identified by typeId is not “EDGE_LOC”.

**longitude** An optional field which, if present, should be a floating-point number that will define the Longitude for the Cache Group.

**name** The Name of the Cache Group

**parentCachegroupId** An optional field which, if present, should be an integer that is the ID of a Parent for this Cache Group.

**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 Cache Group’s Type

Note: The actual, integral, unique identifiers for these Types must first be obtained, generally via types.

---

# 160: Request Example

```
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,
}
```

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.
"name": "test",
"fallbacks": [],
"fallbackToClosest": true,
"shortName": "test",
"typeId": 23,
"localizationMethods": ["GEO"]
}

Response Structure

**fallbacks**  An array of strings that are *Cache Group names* that are registered as *Fallbacks* for this *Cache Group*

New in version ATCv4.0: This field was added to all versions of this endpoint with ATC version 4.0

**fallbackToClosest**  A boolean value that defines the *Fallback to Closest* behavior of this *Cache Group*

**id**  An integer that is the *ID* of the *Cache Group*

**lastUpdated**  The time and date at which this entry was last updated in an ISO-like format

**latitude**  A floating-point *Latitude* for the *Cache Group*

**localizationMethods**  An array of *Localization Methods* as strings

New in version ATCv4.0: This field was added to all versions of this endpoint with ATC version 4.0

**longitude**  A floating-point *Longitude* for the *Cache Group*

**name**  A string containing the *Name* of the *Cache Group*

**parentCachegroupId**  An integer that is the *ID* of this *Cache Group’s Parent* - or null if it doesn’t have a *Parent*

**parentCachegroupName**  A string containing the *Name* of this *Cache Group’s Parent* - or null if it doesn’t have a *Parent*

**secondaryParentCachegroupId**  An integer that is the *ID* of this *Cache Group’s Secondary Parent* - or null if it doesn’t have a *Secondary Parent*

**secondaryParentCachegroupName**  A string containing the *Name* of this *Cache Group’s Secondary Parent Cache Group* - or null if it doesn’t have a *Secondary Parent*

**shortName**  A string containing the *Short Name* of the *Cache Group*

**typeId**  An integral, unique identifier for the ‘*Type*’ of the *Cache Group*

**typeName**  A string that names the *Type* of this *Cache Group*
# 161: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: t1W65/2kj25QyHt01b0xpBaAR2sXu2kOsRZ49WjKZp/...AK5S1YWhX7VNCuUGiN1VNM4QRNq0DC/7ewYDFUncA==
X-Server-Name: traffic_ops_golang/
Date: Wed, 14 Nov 2018 19:14:28 GMT
Content-Length: 385

```json
{
  "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,
    "fallbacks": [],
    "fallbackToClosest": true,
    "localizationMethods": [
      "GEO"
    ],
    "typeName": "EDGE_LOC",
    "typeId": 23,
    "lastUpdated": "2018-11-14 19:14:28+00"
  }
}
```

DELETE

Delete a Cache Group. A Cache Group which has assigned servers or is the Parent of one or more other Cache Groups cannot be deleted.

Auth. Required  Yes
Roles Required  “admin” or “operations”

Response Type  undefined

Request Structure

Table 98: Request Path Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The ID of a Cache Group to be deleted</td>
</tr>
</tbody>
</table>

# 162: 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

# 163: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: 5jZBgO7h1eNF70J/cm1bi3Hf9KJPx+WLMbH/pSKF3FWb/10GUHIN352OB+1N5LYCkmk3izGbfTkiruG8I41Q==
X-Server-Name: traffic_ops_golang/
Date: Wed, 14 Nov 2018 20:31:04 GMT
Content-Length: 57

{ "alerts": [ {
  "text": "cachegroup was deleted.",
  "level": "success"
} ]
```

cachegroups/{{ID}}/deliveryservices

5.1. Traffic Ops API
POST

Assigns all of the “assignable” servers within a Cache Group to one or more Delivery Services.

**Note:** “Assignable” here means all of the Cache Group’s servers that have a Type that matches one of the glob patterns EDGE* or ORG*. If even one server of any Type exists within the Cache Group that is not assigned to the same CDN as the Delivery Service to which an attempt is being made to assign them, the request will fail.

**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’s servers are being assigned

# 164: Request Example

```shell
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/json

{"deliveryServices": [2]}
```

**Response Structure**

deliveryServices An array of integral, unique identifiers for Delivery Services to which the Cache Group’s servers have been assigned

id An integer that is the Cache Group’s ID

serverNames An array of the (short) hostnames of all of the Cache Group’s “assignable” Servers

# 165: 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
```
cachegroups/{{ID}}/parameters

GET

Gets all of a Cache Group’s parameters.

Auth. Required  Yes
Roles Required  None
Response Type  Array
Request Structure

Table 99: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameterId</td>
<td>no</td>
<td>Show only the Parameter with the given ID</td>
</tr>
<tr>
<td>orderBy</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n&quot;th page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

Table 100: Request Path Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The ID of a Cache Group</td>
</tr>
</tbody>
</table>

Response Structure

- `configFile` The Parameter’s Config File
- `id` The Parameter’s ID
- `lastUpdated` The date and time at which this Parameter was last updated, in an ISO-like format
- `name` Name of the Parameter
- `secure` A boolean value describing whether or not the Parameter is Secure
- `value` The Parameter’s Value

# 166: 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: *
```
cachegroups/{{ID}}/queue_update

POST

Queue or “dequeue” updates for all of a Cache Group’s servers, limited to a specific CDN.

Auth. Required  Yes
Roles Required  “admin” or “operations”
Response Type  Object

Request Structure

Table 101: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The ID of 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 Queue Updates, or a “dequeue” thereof |
|        | cdnId The integral, unique identifier for the CDN in need of Queue Updates, or a “dequeue” thereof |

1 Either ‘cdn’ or ‘cdnID’ must be in the request data (but not both).
# 167: 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

{"action": "queue", "cdn": "CDN-in-a-Box"}
```

## Response Structure

**action** The action processed, one of “queue” or “dequeue”

**cachegroupId** An integer that is the ID of the Cache Group for which Queue Updates was performed or cleared

**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 Cache Group’s servers which are also assigned to the CDN specified in the "cdn" field

---

# 168: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: UAcP7Lrf1U1RnfR4UgbQrJczl5rkrcL0tTXJTFvIUXxK1Ekl2kHkE4veWjDaVIhJJ6YQg8jmPGQp
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"
   ],
}}
```
cachegroups/{{id}}/unassigned_parameters

GET

Gets all the Parameters that are not a specific Cache Group’s parameters.

Auth. Required Yes
Roles Required None
Response Type Array

Request Structure

Table 102: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameterId</td>
<td>no</td>
<td>Show only the Parameter with the given ID</td>
</tr>
<tr>
<td>orderBy</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

Table 103: 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 ID of a Cache Group</td>
</tr>
</tbody>
</table>

Response Structure

configFile The Parameter’s Config File
id  The Parameter’s ID

lastUpdated  The date and time at which this Parameter was last updated, in an ISO-like format

name  Name of the Parameter

secure  A boolean value that describes whether or not the Parameter is Secure

value  The Parameter’s Value

# 169: Response Example

```json
{
    "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"
        },
        {
            "lastUpdated": "2018-10-09 11:55:46.014844+00",
            "value": "/opt/trafficserver/etc/trafficserver",
            "secure": false,
            "name": "location",
            "id": 6842,
            "configFile": "hdr_rw_bamtech-nhl-live-t.config"
        },
        {
            "lastUpdated": "2018-10-09 11:55:46.014844+00",
            "value": "/opt/trafficserver/etc/trafficserver",
            "secure": false,
            "name": "location",
            "id": 6843,
            "configFile": "hdr_rw_mid_bamtech-nhl-live-t.config"
        }
    ]
}
```

cachegroups/{{parameter ID}}/parameter/available

Deprecated since version ATCv4.
Danger: 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

ResponseType Array

Request Structure

Table 104: 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 ID of the Parameter of interest</td>
</tr>
</tbody>
</table>

Response Structure

id An integer that is the Cache Group’s ID

name A string that is the Cache Group’s name

# 170: Response Example

```json
{
   "alerts": [
      {
         "level": "warning",
         "text": "This endpoint is deprecated, please use 'GET /cachegroupparameters & GET /cachegroups' instead"
      }
   ],
   "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** A string that is a *Cache Group’s Name*

  # 171: Response Example

```json
{
    "response": [
        {
            "name": "TRAFFIC_ANALYTICS"
        },
        {
            "name": "TRAFFIC_OPS"
        },
        {
            "name": "TRAFFIC_OPS_DB"
        }
    ]
}
```

(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 cache servers 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.

Auth. Required Yes
Roles Required None
Response Type Array

Request Structure

No parameters available.

Response Structure

- **cachegroup** A string that is the Name of the Cache Group to which this cache server belongs
- **connections** Current number of TCP connections maintained by the cache server
**healthy** true if Traffic Monitor has marked the *cache server* as “healthy”, false otherwise

See also:

*Health Protocol*

**hostname** The (short) hostname of the *cache server*

**ip** The IP address of the *cache server*

**kbps** The *cache server*’s upload speed (to clients) in Kilobits per second

**profile** The *Name* of the *Profile* in use by this *cache server*

**status** The status of the *cache server*

# 172: Response Example

```text
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
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=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: DqbLgitanS8q81/qKC1i+ImMiEMF+SW4G9rb79FWdeWcgwFjL810t1TRp1nNNfHV+tajgjyyK+wMHobqVyaNEfA==
Content-Length: 133

{
  "response": [
    {
      "profile": "ALL",
      "connections": 0,
      "ip": null,
      "status": "ALL",
      "healthy": true,
      "kbps": 0,
      "hostname": "ALL",
      "cachegroup": "ALL"
    }
  ]
}
```

**capabilities**
GET

Get all capabilities.

Auth. Required Yes
Roles Required None
Response Type Array

Request Structure

Table 105: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>no</td>
<td>Return only the capability that has this name</td>
</tr>
<tr>
<td>order</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

# 173: Request Example

```bash
get /api/1.4/capabilities?name=test HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

name Name of the capability
description Describes the permissions covered by the capability.
lastUpdated Date and time of the last update made to this capability, in an ISO-like format
# 174: 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
Vary: Accept-Encoding
Transfer-Encoding: chunked
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: c18+GtXZI8PoCSwuAzBh1+6w3vDpKQTa/cDJC0WHxdpguOL378KBxGW5PCSyZfJUb7wPyOL5qKMn6NNTuFhg==
X-Server-Name: traffic_ops_golang/
Date: Thu, 15 Aug 2019 17:20:20 GMT
Content-Length: 161

{
  "response": [
    {
      "description": "This is only a test. If this were a real capability, it might do something",
      "lastUpdated": "2019-08-15 17:18:03+00",
      "name": "test"
    }
  ]
}
```

**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

# 175: Request Example

```
POST /api/1.4/capabilities HTTP/1.1
Host: trafficops.infra.ciab.test
```
Apache Traffic Control Documentation, Release 4.0.0

User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 73
Content-Type: application/json

{
   "name": "testquest",
   "description": "A test capability for API examples"
}

Response Structure

description Describe the permissions covered by the capability.

lastUpdated Date and time of the last update made to this capability, in an ISO-like format

name Name of the capability

# 176: 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-Encoding: gzip
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; Expires=Tue, 07 Jan 2020 20:06:18 GMT; Max-Age=3600; HttpOnly
X-Server-Name: traffic_ops_golang/
Date: Tue, 07 Jan 2020 19:06:18 GMT
Content-Length: 225

{
   "alerts": [
      {
         "text": "Capability created.",
         "level": "success"
      },
      {
         "text": "This endpoint is deprecated, and will be removed in the future",
         "level": "warning"
      }
   ],
   "response": {

(continues on next page)
capabilities/{{name}}

Deprecated since version ATCv4: As of ATC version 4.0, every method of this endpoint is deprecated. See each method for details.

**GET**

Deprecated since version ATCv4: This method of this endpoint is deprecated, please use the ‘name’ parameter of a GET request to *capabilities* instead.

Get a capability by name.

- **Auth. Required** Yes
- **Roles Required** None
- **Response Type** Array

### 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 capability of interest</td>
</tr>
</tbody>
</table>

**# 177: Request Example**

```text
GET /api/1.5/capabilities/testquest HTTP/1.1
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: mojolicious=...
```

### 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

# 178: 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-Encoding: gzip
Content-Length: 221
Content-Type: application/json
Date: Wed, 29 Jan 2020 20:49:49 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 30 Jan 2020 00:49:49 GMT; path=/; HttpOnly
Vary: Accept-Encoding
```

```json
{
    "alerts": [
        {
            "level": "warning",
            "text": "This endpoint is deprecated, please use 'GET /capabilities with the 'name' query parameter' instead"
        }
    ],
    "response": [
        {
            "lastUpdated": "2020-01-29 20:10:53.821978+00",
            "name": "testquest",
            "description": "A test capability for API examples"
        }
    ]
}
```

**PUT**

Deprecated since version ATCv4: This method of this endpoint is deprecated. In the future, Capabilities will be immutable, and so no alternative is offered.

Edit a capability.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Object
## 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 capability to be modified</td>
</tr>
</tbody>
</table>

**description** Describes the APIs covered by the capability

### # 179: Request Example

```
PUT /api/1.5/capabilities/testquest HTTP/1.1
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: mojolicious=...
Content-Length: 36

{"description": "A new 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

### # 180: 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-Encoding: gzip
Content-Length: 224
Content-Type: application/json
Date: Wed, 29 Jan 2020 21:25:10 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 30 Jan 2020 01:25:10 GMT; path=/; HttpOnly
Vary: Accept-Encoding

{ "alerts": [  
```
DELETE

Deprecated since version ATCv4: This method of this endpoint is deprecated. In the future, Capabilities will be immutable, and so no alternative is offered.

Delete a capability.

Auth. Required Yes

Roles Required “admin” or “operations”

Response Type undefined

Request Structure

Table 108: 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>

# 181: Request Example

DELETE /api/1.5/capabilities/testquest HTTP/1.1
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: mojolicious=...
Content-Length: 0
Response Structure

# 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
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-Encoding: gzip
Content-Length: 146
Content-Type: application/json
Date: Wed, 29 Jan 2020 21:27:57 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Thu, 30 Jan 2020 01:27:57 GMT; path=/; HttpOnly
Vary: Accept-Encoding

{ "alerts": [
  {
    "level": "success",
    "text": "Capability deleted."
  },
  {
    "level": "warning",
    "text": "This endpoint and its functionality is deprecated, and will be removed in the future"
  }
]}
```

cdns

Extract information about all CDNs

GET

Auth. Required  Yes
Roles Required  None
Response Type  Array
Request Structure

Table 109: Request Query Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>orderBy</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n&lt;sup&gt;th&lt;/sup&gt; page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</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

# 183: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: z9P1NkxGebPncUhaChDhtYKYI+XVZfhE6Y84TuwoASZFIMfISELwADlpvpPTN+wwnzBfREksLYn+03RX9w==
X-Server-Name: traffic_ops_golang/
Date: Wed, 14 Nov 2018 20:46:57 GMT
Content-Length: 237

```json
{ "response": [...
```
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

# 184: 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

### # 185: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: 1rZRlOfQioGRRrEb4nCfjGGx7y3Ub2h7BZ4z6Nhcy4acPs1KSUNM8QLjWTVwLU4WpkfJNxcOyY8NlK
X-Server-Name: traffic_ops_golang/
Date: Wed, 14 Nov 2018 20:49:28 GMT
Content-Length: 174

{
  "alerts": [
    {
      "text": "cdn was created.",
      "level": "success"
    }
  ],
  "response": {
    "dnssecEnabled": false,
    "domainName": "quest",
    "id": 3,
    "lastUpdated": "2018-11-14 20:49:28+00",
    "name": "test"
  }
}
```

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

---

5.1. Traffic Ops API 379
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

# 186: Response Example

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

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)

1 Following IEEE 754
Request Structure

Table 110: 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.

# 187: Response Example

```

cond %{REMAP_PSEUDO_HOOK}
set-conn-dscp 8 [L]
```

cdns/configs

Deprecated since version ATCv4: Use the GET method of cdns instead.

Danger: This endpoint does not appear to work, and thus its use is strongly discouraged!

GET

Retrieves CDN configuration information.

Auth. Required  Yes

Roles Required  None

Response Type  Array

Request Structure

No parameters available.

5.1. Traffic Ops API 381
Response Properties

- **id** The integral, unique identifier for this CDN
- **name** The CDN’s name

# 188: Response Example

```json
{
    "response": [
        {
            "id": 1,
            "name": "ALL"
        },
        {
            "id": 2,
            "name": "CDN-in-a-Box"
        }
    ],
    "alerts": [
        {
            "text": "This endpoint is deprecated, please use GET /cdns instead",
            "level": "warning"
        }
    ]
}
```

cdns/dnsseckeys/refresh

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**

# 189: 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
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=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: Uwl+924m6Ye3NraFP+RBp1dkhcNTTDyXHZbzRaYVV95p9tP56Z61gckeKSrl0QIkNXjXcCsDN5Dmum7
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 TLD assigned to this CDN

**parameterId**  The ID 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 ID of 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

# 190: Response Example

```json
{
  "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

```
# 191: Response Example
{
  "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 111: 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

```json
{ "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 112: 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

# 193: Request Example

```
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

# 194: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
```
DELETE

Allows a user to delete a specific CDN

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>ID</td>
<td>The integral, unique identifier of the CDN to delete</td>
</tr>
</tbody>
</table>

Response Structure

# 195: 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
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

# 196: 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
```
Response Structure

- **action** The action processed, either "queue" or "dequeue"
- **cdnId** The integral, unique identifier for the CDN on which *Queue Updates* was performed or cleared

---

# 197: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: rBpFfrrP+9IFkwsR1oEM+v+I8MuBZDxqFu+WUTGtRGypnAn2gHooPoNQRyVvJGjyIqrLXlvqjEtv+
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

**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 115: 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 <em>Snapshot</em> shall be taken</td>
</tr>
</tbody>
</table>

# 198: 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

# 199: 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: z4PhNX7vuL3xVChQ1m2AB9Yg5AULVxXcg/Sp1dNs6c5H0NE8XYysP+DGKHyuwvY7kxvUdBeoGl0DJ6+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}}
```

Danger: 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 Issue #2309 for more information.

5.1. Traffic Ops API  391
GET

Retrieves Edge-tier metrics of one or all Cache Groups.

Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

Table 116: 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

# 200: Response Example

```json
{ "response": [ { "stats": { "count": 1, "98thPercentile": 1668.03, 
```

(continues on next page)
"min": 1668.03,
"max": 1668.03,
"5thPercentile": 1668.03,
"95thPercentile": 1668.03,
"mean": 1668.03,
"sum": 1668.03
},
"data": [
[1425135900000,
1668.03
],
[1425136200000,
nul
]
],
"label": "Origin TPS"
}}

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>
Response Structure

**cacheGroups** An array of objects representing each of the *Cache Groups* being monitored within this CDN

**coordinates** An object representing the geographic location of this *Cache Group*

**latitude** This *Cache Group’s latitude* as a floating-point number

**longitude** This *Cache Group’s longitude* as a floating-point number

**name** A string that is this *Cache Group’s name*

**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 *Snapshot* 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 A string that is the Delivery Service’s XMLID

profiles An array of the Profiles in use by the cache servers and Delivery Services belonging to this CDN

name A string that is 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 also: uptime(1)

  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 A string that names the Profile’s Type

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 IPv4 (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 IPv4 address of this Traffic Monitor

port The port on which this Traffic Monitor listens for incoming connections

profile A string that is 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 cache servers being monitored within this CDN

cacheGroup The Cache Group to which this cache server belongs

fqdn An FQDN that resolves to the cache server’s IPv4 (or IPv6) address

hashId The (short) hostname 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 server’s HTTP proxy

ip6 The cache server’s IPv6 address - when applicable

ip The cache server’s IPv4 address

port The port on which the cache server listens for incoming connections

profile A string that is the Name of the Profile assigned to this cache server

status The status of the cache server

type A string that names the Type of the cache server - should (ideally) be either EDGE or MID

# 201: 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
```
5.1. Traffic Ops API

```json
{
  "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
    }
  },
  {
    "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}\",
      "health.threshold.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}\",
      "health.threshold.availableBandwidthInKbps": ">1750000",
      "health.threshold.loadavg": "25.0",
      "health.threshold.queryTime": 1000,
      "history.count": 30
    }
  }
]
cdns/{{name}}/configs/routing

Deprecated since version ATCv4: Use the GET method of cdns/{{name}}/snapshot instead.

GET

Retrieves CDN routing information.

Auth. Required  Yes

Roles Required  None

Response Type

Request Structure

Table 118: 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>
# 202: Request Example

```
GET /api/1.5/cdns/CDN-in-a-Box/configs/routing HTTP/1.1
User-Agent: curl/7.29.0
Host: trafficops.infra.ciab.test
Accept: */*
Cookie: mojolicious=...
```

Response Structure

- **cacheGroups** A collection of objects that represent *Cache Groups*.
  - **coordinates** An object that represents the geographic location of the *Cache Group*
    - **latitude** number
    - **longitude** number
  - **name** string

- **config** 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

# 203: 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, 27 Jan 2020 19:20:14 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Mon, 27 Jan 2020 23:20:14 GMT;
        path=/; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: Dxgtd9e67IRb9HyhPxG94zijfpCB44mdstlf5ZXokCQoAUKbcPaTu2szPMgineWmNvWxAfgrXo0ZVU
Transfer-Encoding: chunked

{
    "alerts": [
        {
            "level": "warning",
            "text": "This endpoint is deprecated, please use 'GET /cdns/{{name}}/snapshot' instead"
        }
    ],
    "response": {
        "trafficServers": [
            {
                "profile": "ATS_MID_TIER_CACHE",
                "ip": "172.16.239.5",
                "status": "REPORTED",
                "cacheGroup": "CDN_in_a_Box_Mid",
                "ip6": "fc01:9400:1000:8::5",
                "port": 80,
                "deliveryServices": [],
                "hostName": "mid",
                "fqdn": "mid.infra.ciab.test",
                "interfaceName": "eth0",
                "type": "MID",
                "hashId": "mid"
            },
            {
                "profile": "ATS_EDGE_TIER_CACHE",
                "ip": "172.16.239.5",
                "status": "REPORTED",
                "cacheGroup": "CDN_in_a_Box_Mid",
                "ip6": "fc01:9400:1000:8::5",
                "port": 80,
                "deliveryServices": [],
                "hostName": "mid",
                "fqdn": "mid.infra.ciab.test",
                "interfaceName": "eth0",
                "type": "MID",
                "hashId": "mid"
            }
        ]
    }
}
```
"ip": "172.16.239.4",
"status": "REPORTED",
"cacheGroup": "CDN_in_a_Box_Edge",
"ip6": "fc01:9400:1000:8::4",
"port": 80,
"deliveryServices": [],
"hostName": "edge",
"fqdn": "edge.infra.ciab.test",
"interfaceName": "eth0",
"type": "EDGE",
"hashId": "edge"
},
"stats": {
"trafficOpsPath": "/api/1.5/cdns/CDN-in-a-Box/configs/routing",
"cdnName": "CDN-in-a-Box",
"trafficOpsVersion": "4.0.0-10449.03d91ae3.el7",
"trafficOpsUser": "admin",
"date": 1580152814,
"trafficOpsHost": "trafficops.infra.ciab.test"
},
"cacheGroups": [
{
"coordinates": {
"longitude": -77.036574,
"latitude": 38.897663
},
"name": "CDN_in_a_Box_Edge"
},
{
"coordinates": {
"longitude": -77.036574,
"latitude": 38.897663
},
"name": "CDN_in_a_Box_Mid"
}
],
"config": {
"tld.soa.admin": "twelve_monkeys",
"dnssec.dynamic.response.expiration": "300s",
"api.cache-control.max-age": 10,
"zonemanager.threadpool.scale": "0.50",
"coveragezone.polling.interval": 3600000,
"zonemanager.polling.interval": 18000000
}
"federationmapping.polling.interval": 60000,
"steeringmapping.polling.interval": 60000,
"tld.ttls.DNSKEY": 30,
"geolocation.polling.interval": 86400000,
"tld.soa.expire": 604800,
"federationmapping.polling.url": "https://
˓→{toHostname}/internal/api/1.3/federations.json",
"coveragezone.polling.url": "https://
˓→trafficops.infra.ciab.test:443/coverage-zone.json",
"tld.soa.minimum": 30,
"geolocation.polling.url": "https://
˓→trafficops.infra.ciab.test:443/GeoLite2-City.mmdb.gz",
"keystore.maintenance.interval": 300,
"zonemanager.cache.maintenance.interval": 300,
"domain_name": "mycdn.ciab.test",
"tld.ttls.AAAA": 3600,
"tld.soa.refresh": 28800,
"consistent.dns.routing": "true",
"tld.ttls.SOA": 86400,
"neustar.polling.interval": 86400000,
"tld.ttls.NS": 3600,
"tld.ttls.DS": 30,
"certificates.polling.interval": 300000,
"tld.ttls.A": 3600,
"tld.soa.retry": 7200
],
"trafficMonitors": [
{
"profile": "RASCAL-Traffic_Monitor",
"location": "CDN_in_a_Box_Edge",
"ip": "172.16.239.11",
"status": "ONLINE",
"ip6": "fc01:9400:1000:8::b",
"port": 80,
"hostName": "trafficmonitor",
"fqdn": "trafficmonitor.infra.ciab.
˓→test"
},
"trafficRouters": [
{
"profile": "CCR_CIAB",
"location": "CDN_in_a_Box_Edge",
"ip": "172.16.239.12",
"status": "ONLINE",
"secureApiPort": 3333,
"ip6": "fc01:9400:1000:8::c",
"port": 80,
}]}
"hostName": "trafficrouter",
"fqdn": "trafficrouter.infra.ciab.test",
"apiPort": 3333
}
}
]
}
}

cdns/{{name}}/dnsseckeys/ksk/generate

New in version 1.4.

POST

Generates a new KSK for a specific CDN.

Auth. Required Yes
Roles Required “admin”
Response Type Object (string)

Request Structure

Table 119: 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>

expirationDays The integral number of days until the newly generated KSK expires

effectiveDate An optional string containing the date and time at which the newly generated KSK becomes effective, in RFC 3339 format. Defaults to the current time if not specified

Response Structure

# 204: Response Example

{ "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 120: 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>

Table 121: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>order by</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>
<tr>
<td>sort Order</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>Return the n&lt;sup&gt;th&lt;/sup&gt; page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

# 205: Request Example

```
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

### # 206: 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+7G5KcOfCtnE3Dq5DCobWtGRUKSpqIDkflZoG5+paq4Elaj2GqUb6vGVsd+TpPg75ML1hyqfdfl
x-server-name: traffic_ops_golang/
content-length: 170
date: Wed, 05 Dec 2018 00:35:40 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"
      }
    }
  ]
}
```

**POST**

Creates a new federation.
Auth. Required  Yes
Roles Required  “admin”
Response Type  Object

Request Structure

Table 122: 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

description  An optional description of the federation
ttl  Time to Live (TTL) for the name record used for cname

# 207: 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

# 208: 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: rRsWAIhXzV1j8Hy+8aFjp4Jo1QGTK49m0N1AP5QDyyAZlTfNiIdgtcgiehu7FiN1IPWRFiv6D9CygF
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.",      "ttl": 48,      "description": "A test federation",      "lastUpdated": "2018-12-05 00:05:16+00"    }}
```

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 123: 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>

Table 124: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>orderBy</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>
<tr>
<td>sortOrder</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (”desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

# 209: 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

# 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: SJA7G+7G5KcOfCtnE3Dq5DCobWtGRUKSpipDKfLZoG5+paq4Ela2GqUb6vGvsd+TpPg75ML1hyqfdF
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": "demol"
    }
  }
]}
```

PUT

Updates a federation.

Auth. Required  Yes

Roles Required  “admin”

Response Type  Object
Request Structure

Table 125: 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 c.name

# 211: 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 c.name, in hours
# 212: 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: qcjfQ+gDjNXQ1aq+d1ddgrkFkFyxsFF+SHDqgH0vVHBVksmU0aTFgltozek/
   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 126: 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>
# 213: 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

# 214: 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: Cnkfj6dmzTD3if9oiDq33tqf7CnAflKK/SPgqJyfu6HUf0jLJOGKIZvkcs2wWY6EjLVdw5qsatsd4FPoCyjvcw=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 127: 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>

# 215: Request Example

GET /api/1.4/cdns/CDN-in-a-Box/health HTTP/1.1
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: mojolicious=...

Response Structure

- **cachegroups** An array of objects describing the health of each *Cache Group*
  - **name** A string that is the *Cache Group’s Name*
  - **offline** The number of OFFLINE *cache servers* in the *Cache Group*
  - **online** The number of ONLINE *cache servers* in the *Cache Group*
  - **totalOffline** Total number of OFFLINE *cache servers* across all *Cache Groups* which are assigned to the CDN defined by the *name* request path parameter
  - **totalOnline** Total number of ONLINE *cache servers* across all *Cache Groups* which are assigned to the CDN defined by the *name* request path parameter

# 216: Response Example

```text
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-Encoding: gzip
Content-Length: 108
Content-Type: application/json
Date: Tue, 03 Dec 2019 21:33:59 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Wed, 04 Dec 2019 01:33:59 GMT;
```
cdns/name/{{name}}

GET

Extract information about a CDN, identified by name.

**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>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
# 217: 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: bTz86xdnGfbKhxnneb4geXohaw3lhG+h5wc2l/ncHFATwp1h80h+txxySCIVfa0hgBrJHEdpGZQsH5w5IknsrQ==
X-Server-Name: traffic_ops_golang/
Date: Wed, 14 Nov 2018 21:22:16 GMT
Content-Length: 137
```

```json
{
    "response": [
        {
            "dnssecEnabled": false,
            "domainName": "mycdn.ciab.test",
            "id": 2,
            "lastUpdated": "2018-11-14 18:21:14+00",
            "name": "CDN-in-a-Box"
        }
    ]
}
```

**DELETE**

Allows a user to delete a CDN by name

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** undefined

### Request Structure

**Table 129: 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 deleted</td>
</tr>
</tbody>
</table>

### Response Structure
# 218: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: Zy4cJN6BEct4ltFLN4e296mM8XnzOsOEQ3/jp4TA3L+g8qtkI0WrL+ThcFq4xbJPU+KHVDi+sB0JBav3xsYPqQ==
X-Server-Name: traffic_ops_golang/
Date: Wed, 14 Nov 2018 20:59:22 GMT
Content-Length: 58

```json
{
  "alerts": [
    {
      "text": "cdn was deleted.",
      "level": "success"
    }
  ]
}
```

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 130: 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>

5.1. Traffic Ops API
Response Structure

**name**  The name of the CDN or *Delivery Service* to which the enclosed keys belong

**zsk**  The short-term 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 KSK

**dsRecord**  An optionally present object containing information about the algorithm used to generate the key

New in version 1.2.

**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

# 219: Response Example

```json
{
  "response": {
    "cdn1": {
      "zsk": {
        "ttl": "60",
        "inceptionDate": "1426196750",
        "private": "zsk private key",
        "public": "zsk public key",
        "expirationDate": "1428788750",
        "name": "foo.kabletown.com."
      },
      "ksk": {
        "name": "foo.kabletown.com.",
        "expirationDate": "1457732750",
        "public": "ksk public key",
        "private": "ksk private key",
        "inceptionDate": "1426196750",
        "ttl": "60",
        "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

5.1. Traffic Ops API

421
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 131: 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

# 220: 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 132: 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** A string that is the *xml_id* of the *Delivery Service* using the SSL key within *certificate*

---

# 221: Response Example

```
{
  "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 133: 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>

# 222: Request Example

```plaintext
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

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.dynamic.concurrencylevel  An integer that defines the size of the concurrency level (threads) of the Guava cache used by ZoneManager to store zone material

dnssec.dynamic.initialcapacity  An integer that defines the initial size of the Guava cache, default is 10000. Too low of a value can lead to expensive resizing

dnssec.init.timeout  An integer that defines the number of minutes to allow for zone generation, this bounds the zone priming activity

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  A string that is the TLD served by the CDN

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 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 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 via HTTP

secure.api.port  An optionally present string containing the port number on which the Traffic Router API is served by this Traffic Router via HTTPS

fqdn  This Traffic Router’s 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  A string which is 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 servers in the CDN; the values corresponding to those keys are routing information for said servers

cacheGroup  A string that is 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 (Fully Qualified Domain Names) that resolve to this cache server

---

Note:  Only Edge-tier cache servers can be assigned to a Delivery Service, and therefore this field will only be present when type is "EDGE".

---

fqdn  The server’s 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 to 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; which ought to be one of (but in practice need not be in certain special circumstances):

  **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_ids of all of the Delivery Services within the CDN

**anonymousBlockingEnabled**  A string containing a boolean that tells whether or not Anonymous Blocking is set on 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*
consistentHashQueryParameters A set of query parameters that Traffic Router should consider when determining a consistent hash for a given client request.

New in version ATCv4: This field was added to all versions of this endpoint with ATC version 4.0.

consistentHashRegex An optional regular expression that will ensure clients are consistently routed to a cache server based on matches to it.

New in version ATCv4: This field was added to all versions of this endpoint with ATC version 4.0.

coverageZoneOnly A string containing a boolean that tells whether or not this Delivery Service routes traffic based only on its Coverage Zone File

See also:

Geo Limit

depthCachingType A string that defines the Deep Caching setting of this Delivery Service

dispersion An object describing the “dispersion” - or number of cache servers within a single Cache Group across which the same content is spread - within the Delivery Service

- limit The maximum number of cache servers in which the response to a single request URL will be stored

Note: If this is greater than the number of cache servers 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 cache servers chosen for content dispersion are chosen randomly or based on a consistent hash of the request URL; one of:

- “false” Cache servers will be chosen consistently
- “true” Cache servers will be chosen at random

domains An array of domains served by this Delivery Service
**ecsEnabled** A string containing a boolean from *EDNS0 Client Subnet Enabled* that tells whether EDNS0 client subnet is enabled on this *Delivery Service*; one of:

- **“false”** EDNS0 client subnet is not enabled on this *Delivery Service*
- **“true”** EDNS0 client subnet is enabled on 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 defines the *IPv6 Routing Enabled* setting for 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 name of 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, or the name requested for resolution in a DNS 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

---

1 These only apply to HTTP-routed *Delivery Services*
**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* (and/or possibly the *Deep Coverage Zone File*) and the IP-to-geographic-location database

- **lat** Geographic latitude as a floating point number
- **long** Geographic longitude as a floating point number

**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

See also:

*Protocol*

**regionalGeoBlocking** A string containing a boolean that defines the *Regional Geoblocking* setting of this *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 A string that is this Delivery Service's Routing Name

soa An object defining the SOA record for the Delivery Service’s TLDs (Top-Level Domains) (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 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
See also:

Protocol

**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.

**edgeLocations** An object containing keys which are the names of Edge-Tier *Cache Groups* within the CDN

**backupLocations** An object that describes this *Cache Group’s Fallbacks*

**fallbackToClosest** A string containing a boolean which defines the *Fallback to Closest* behavior of this *Cache Group*; one of:

- **“false”** Do not fall back on the closest available *Cache Group*
- **“true”** Fall back on the closest available *Cache Group*

**list** If this *Cache Group* has any Fallbacks, this key will appear and will be an array of those *Cache Groups’ Names*

**latitude** A floating point number that defines this *Cache Group’s Latitude*

**localizationMethods** An array of strings that represents this *Cache Group’s Localization Methods*

**longitude** A floating point number that defines this *Cache Group’s Longitude*

**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** A string which is 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  A string which is 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

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 this Cache Group’s Fallbacks

fallbackToClosest  A string containing a boolean which defines this Cache Group’s Fallback to Closest setting; one of:

“false”  Do not fall back on the closest available Cache Group

“true”  Fall back on the closest available Cache Group

latitude  A floating point number that defines this Cache Group’s Latitude

localizationMethods  An array of strings that represents this Cache Group’s Localization Methods
longitude A floating point number that defines this Cache Group’s Longitude

# 223: 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: 220bc4XXwaj+s7ODd3QAF5leGj061nApiN5E8H/B2RgxSphnQIfnwy6WwhBDjonWXPV11WDCj8MO+rR+1AabMg==
X-Server-Name: traffic_ops_golang/
Date: Wed, 12 Dec 2018 17:36:25 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://trafficopsinfra.ciab.test:443/coverage-zone.json",
            "dnssec.dynamic.response.expiration": "300s",
            "dnssec.enabled": "false",
            "domain_name": "mycdn.ciab.test",
            "federationmapping.polling.interval": "60000",
            "federationmapping.polling.url": "https://_traffic_ops_golang/
            "geolocation.polling.interval": "86400000",
            "geolocation.polling.url": "https://trafficopsinfra.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",
            }
        },
```

(continues on next page)
"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",
    "locationId": "CDN_in_a_Box_Mid",
    "port": 80,
    "profile": "ATS_MID_TIER_CACHE",
    "status": "REPORTED",
    "type": "MID",
    "routingDisabled": 0
  }
},
"contentRouters": {

"trafficrouter": {
    "api.port": "3333",
    "secure.api.port": "3443",
    "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": {
    "demo1": {
        "anonymousBlockingEnabled": "false",
        "coverageZoneOnly": "false",
        "dispersion": {
            "limit": 1,
            "shuffled": "true"
        },
        "domains": [
            "demo1.mycdn.ciab.test"
        ],
        "geolocationProvider": "maxmindGeolocationService",
        "matchsets": [
            {
                "protocol": "HTTP",
                "matchlist": [
                    {
                        "regex": ".*\.demo1\..*",
                        "match-type": "HOST"
                    }
                ]
            }
        ],
        "missLocation": {
            "lat": 42,
            "long": -88
        },
        "protocol": {
            "acceptHttps": "false",
            "redirectToHttps": "false"
        },
        "regionalGeoBlocking": "false",
        "soa": {
            "lat": 42,
            "long": -88
        }
    }
}
"admin": "traffic_ops",
"expire": "604800",
"minimum": "30",
"refresh": "28800",
"retry": "7200"
},
"sslEnabled": "false",
"ttls": {
   "A": "",
   "AAAA": "",
   "NS": "3600",
   "SOA": "86400"
},
"ip6RoutingEnabled": "true",
"ecsEnabled": "false",
"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",
         "CZ",
         "DEEP_CZ"
      ]
   }
},
"monitors": {
   "trafficmonitor": {
      "
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 134: 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>

# 224: Request Example

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
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

- **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.dynamic.concurrencylevel** An integer that defines the size of the concurrency level (threads) of the Guava cache used by ZoneManager to store zone material

- **dnssec.dynamic.initialcapacity** An integer that defines the initial size of the Guava cache, default is 10000. Too low of a value can lead to expensive resizing

- **dnssec.init.timeout** An integer that defines the number of minutes to allow for zone generation, this bounds the zone priming activity
**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**  A string that is the TLD served by the CDN

**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 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 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 via HTTP

secure.api.port An optionally present string containing the port number on which the Traffic Router API is served by this Traffic Router via HTTPS

fqdn This Traffic Router’s 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 A string which is 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

ccontentServers An object containing keys which are the (short) hostnames of the Edge-Tier cache servers in the CDN; the values corresponding to those keys are routing information for said servers

cacheGroup A string that is 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 servers can be assigned to a Delivery Service, and therefore this field will only be present when type is "EDGE".

fqdn The server’s 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 to 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; which ought to be one of (but in practice need not be in certain special circumstances):
- 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_ids of all of the Delivery Services within the CDN.

anonymousBlockingEnabled A string containing a boolean that tells whether or not Anonymous Blocking is set on 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

consistentHashQueryParameters A set of query parameters that Traffic Router should consider when determining a consistent hash for a given client request.

New in version ATCv4: This field was added to all versions of this endpoint with ATC version 4.0.

consistentHashRegex An optional regular expression that will ensure clients are consistently routed to a cache server based on matches to it.
New in version ATCv4: This field was added to all versions of this endpoint with ATC version 4.0.

coverageZoneOnly  A string containing a boolean that tells whether or not this Delivery Service routes traffic based only on its Coverage Zone File

See also:

Geo Limit

deepCachingType  A string that defines the Deep Caching setting of this Delivery Service

dispersion  An object describing the “dispersion” - or number of cache servers within a single Cache Group across which the same content is spread - within the Delivery Service

limit  The maximum number of cache servers in which the response to a single request URL will be stored

Note: If this is greater than the number of cache servers 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 cache servers chosen for content dispersion are chosen randomly or based on a consistent hash of the request URL; one of:

“false” Cache servers will be chosen consistently

“true” Cache servers will be chosen at random

domains  An array of domains served by this Delivery Service

cesEnabled  A string containing a boolean from EDNS0 Client Subnet Enabled that tells whether EDNS0 client subnet is enabled on this Delivery Service; one of:

“false” EDNS0 client subnet is not enabled on this Delivery Service

“true” EDNS0 client subnet is enabled on 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 defines the IPv6 Routing Enabled setting for 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 name of 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, or the name requested for resolution in a DNS 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

**missLocation** An object representing the default geographic coordinates to use for a client when lookup of their IP

---

1 These only apply to HTTP-routed Delivery Services
has failed in both the *Coverage Zone File* (and/or possibly the *Deep Coverage Zone File*) and the IP-to-geographic-location database

- **lat** Geographic latitude as a floating point number
- **long** Geographic longitude as a floating point number

**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

See also:

*Protocol*

**regionalGeoBlocking** A string containing a boolean that defines the *Regional Geoblocking* setting of this *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** A string that is this *Delivery Service’s Routing Name*

**soa** An object defining the 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 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*

---

**See also:**

*Protocol*

**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.

**edgeLocations** An object containing keys which are the names of Edge-Tier Cache Groups within the CDN

**backupLocations** An object that describes this Cache Group’s Fallbacks

**fallbackToClosest** A string containing a boolean which defines the Fallback to Closest behavior of this Cache Group; one of:

- **false**” Do not fall back on the closest available Cache Group
- **true**” Fall back on the closest available Cache Group

**list** If this Cache Group has any Fallbacks, this key will appear and will be an array of those Cache Groups’ Names

**latitude** A floating point number that defines this Cache Group’s Latitude

**localizationMethods** An array of strings that represents this Cache Group’s Localization Methods

**longitude** A floating point number that defines this Cache Group’s Longitude

**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** A string which is 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** A string which is 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
- **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 this *Cache Group’s Fallbacks*
  - **fallbackToClosest** A string containing a boolean which defines this *Cache Group’s Fallback to Closest* setting; one of:
    - **“false”** Do not fall back on the closest available *Cache Group*
    - **“true”** Fall back on the closest available *Cache Group*
  - **latitude** A floating point number that defines this *Cache Group’s Latitude*
  - **localizationMethods** An array of strings that represents this *Cache Group’s Localization Methods*
  - **longitude** A floating point number that defines this *Cache Group’s Longitude*
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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: MWzgAYngmUlIEIxRa0C6VfY+yMMu7T90C1Ij1Aul58pA7J7DiS6r8wjVRVW8W2Eu2V9BC70EacRlf
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",
    "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": {
            "demol": [
                "edge.demol.mycdn.ciab.test"
            ],
            "routingDisabled": 0
        },
        "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",
        "locationId": "CDN_in_a_Box_Mid",
        "port": 80,
        "profile": "ATS_MID_TIER_CACHE",
        "status": "REPORTED",
        "type": "MID",
        "routingDisabled": 0
    }
},
"contentRouters": {
    "trafficrouter": {
        "api.port": "3333",
        "routingDisabled": 0
    }
}
"secure.api.port": "3443",
"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": {
  "demo1": {
    "anonymousBlockingEnabled": "false",
    "coverageZoneOnly": "false",
    "dispersion": {
      "limit": [],
      "shuffled": "true"
    },
    "domains": [
      "demo1.mycdn.ciab.test"
    ],
    "geolocationProvider": "maxmindGeolocationService",
    "matchsets": [
    {
      "protocol": "HTTP",
      "matchlist": [
        {
          "regex": ".*/\d+.demo1/.*",
          "match-type": "HOST"
        }
      ]
    },
    "missLocation": {
      "lat": 42,
      "long": -88
    },
    "protocol": {
      "acceptHttps": "false",
      "redirectToHttps": "false"
    },
    "regionalGeoBlocking": "false",
    "soa": {
      "admin": "traffic_ops",
      "expire": "604800",
    }
  }
}
"minimum": "30",
"refresh": "28800",
"retry": "7200"
},
"sslEnabled": "false",
"ttl": {
  "A": "",
  "AAAA": "",
  "NS": "3600",
  "SOA": "86400"
},
"ip6RoutingEnabled": "true",
"ecsEnabled": "false",
"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": {
cdns/routing

GET

Retrieves the aggregated routing percentages across all CDNs. This is accomplished by polling stats from all online Traffic Routers via the `/crs/stats` route.

Auth. Required  Yes

Roles Required  None

Response Type  Object

Request Structure

No parameters available

Response Structure

- **cz** The percent of requests to online Traffic Routers that were satisfied by a *Coverage Zone File*
- **deepCz** The percent of requests to online Traffic Routers that were satisfied by a *Deep Coverage Zone File*
dsr  The percent of requests to online Traffic Routers that were satisfied by sending
the client to an overflow Delivery Service

err  The percent of requests to online Traffic Routers that resulted in an error

fed  The percent of requests to online Traffic Routers that were satisfied by sending
the client to a federated CDN

geo  The percent of requests to online Traffic Routers that were satisfied using 3rd
party geographic IP mapping

miss  The percent of requests to online Traffic Routers that could not be satisfied

regionalAlternate  The percent of requests to online Traffic Routers that were satisfied
by sending the client to the alternate, Regional Geo-blocking URL

regionalDenied  The percent of requests to online Traffic Routers that were denied
due to geographic location policy

staticRoute  The percent of requests to online Traffic Routers that were satisfied
with Static DNS Entries

# 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: Wed, 14 Nov 2018 21:29:32 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019
17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512:
7LjytwKyRzSRM4cRIo14OMIJxApFpTWJaSK73rblUIq6ASZjI64XxLVvZP00GRU7XeJ22YKUyQ30q5
Content-Length: 130

{
  "response": {
    "cz": 79,
    "deepCz": 0.50,
    "dsr": 0,
    "err": 0,
    "fed": 0.25,
    "geo": 20,
    "miss": 0.25,
    "regionalAlternate": 0,
    "regionalDenied": 0,
    "staticRoute": 0
  }
}
cdns/usage/overview

New in version 1.2.

Deprecated since version ATCv4: This endpoint and its functionality is deprecated, and will be removed in the future.

**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.

# 227: Response Example

```json
{
    "alerts": [
        {
            "level": "warning",
            "text": "This endpoint and its functionality is deprecated, and will be removed in the future"
        }
    ],
    "response": {
        ...
    }
}
```

(continues on next page)
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 regular expression 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

# 228: 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 \w","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

```
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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512:QMDFOnUfgH4TcZ4YnUQyqnXDier0YiUMIfwBGDCt7ySjw9uASBGsLQW35lKFi4as0vY1HuSSGpe4
X-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 135: 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 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>
<tr>
<td>order</td>
<td>no</td>
<td>Choose the ordering of the results - must be the name of one of the fields of the response array</td>
</tr>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n(^{th}) page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</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

# 230: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: Y2vxC3hpixG6aRNBBT7i2hbAViIJP+dJoqHIzu3acFM+vGay/J5E+eZYOC9RY8hcJPrKNXysZOD8D0b9KsPgaw==
X-Server-Name: traffic_ops_golang/
Date: Wed, 14 Nov 2018 21:32:28 GMT
```
Content-Length: 942

```json
{
  "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"
    },
    {
      "id": 5,
      "name": "from_cachegroup_TRAFFIC_STATS",
      "latitude": 38.897663,
      "longitude": -77.036574,
      "lastUpdated": "2018-10-24 16:07:04+00"
    },
    {
      "id": 6,
      "name": "from_cachegroup_CDN_in_a_Box_Mid",
      "latitude": 38.897663,
      "longitude": -77.036574,
      "lastUpdated": "2018-10-24 16:07:04+00"
    },
    {
      "id": 7,
      "name": "from_cachegroup_CDN_in_a_Box_Edge",
      "latitude": 38.897663,
      "lastUpdated": "2018-10-24 16:07:04+00"
    }
  ]
}
```

(continues on next page)
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])

# 231: Request Example

```
POST /api/1.4/coordinates HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 47
Content-Type: application/json

{"name": "test", "latitude": 0, "longitude": 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
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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: 7pWde2yIIXE1P7o/JVon+5eSCbDw+FGamAzdXzWHXJ81hF+Vh/+tWFckzHYw3rP2kBfvZ+gqLffjQpBCMjt7A==
X-Server-Name: traffic_ops_golang/
Date: Thu, 15 Nov 2018 17:48:55 GMT
Content-Length: 165

```json
{
  "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>
<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>

5.1. Traffic Ops API
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])

# 233: Request Example

```
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

# 234: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: zd03Uvbnv8EbSZZ75Xp5tnnYStZsZTdyPxXnoqK4QZ5WhELLPL8lH1RfoaLTbrUWUeJ8ue2HRz6aBixCCGA==
X-Server-Name: traffic_ops_golang/
Date: Thu, 15 Nov 2018 17:54:30 GMT
Content-Length: 166
```
(continues on next page)
DELETE

Deletes a coordinate

Auth. Required Yes
Roles Required “admin” or “operations”
Response Type undefined

Request Structure

Table 137: 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

# 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: 82x/Wdckqgk4LN5LI1ZfBJ26xkDrUVUGDjs5QFa/Lzap7dU3O2kjav8XW41xeFYj8PDmxHIpb7hiVObvLaxnEDA==

(continues on next page)
current_stats

An API endpoint that returns current statistics for each CDN and an aggregate across them all. New in version 1.2.

GET

Retrieves current stats for each CDN. Also includes aggregate stats across them.

Auth. Required Yes
Roles Required None
Response Type Array

Request Structure

No parameters available.

# 236: Request Example

GET /api/1.4/current_stats HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...

Response Structure

    cdn  The name of the CDN
    connections  Current number of TCP connections maintained
    capacity  85 percent capacity of the CDN in Gps
**bandwidth**  The total amount of bandwidth in Gbs

**Note:** If cdn name is total and capacity is omitted it represents the aggregate stats across CDNs

---

### # 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-Encoding: gzip
Content-Type: application/json
Set-Cookie: mojolicious=; Path=/; HttpOnly
Whole-Content-Sha512: Rs3wgd7v5dP0bQqs4I3J1q6mnWlMSM2AKSAWlrK1kymvDYoFISArF7Kkpygy10I34ym7FtFdMh6U7
X-Server-Name: traffic_ops_golang/
Date: Thu, 14 Nov 2019 15:35:31 GMT
Content-Length: 138

{"response": {
    "currentStats": [
      {
        "bandwidth": null,
        "capacity": null,
        "cdn": "ALL",
        "connections": null
      },
      {
        "bandwidth": 0.000104,
        "capacity": 17,
        "cdn": "CDN-in-a-Box",
        "connections": 4
      },
      {
        "bandwidth": 0.000104,
        "cdn": "total",
        "connections": 4
      }
    ]
}}
```
dbdump

**Caution:** This is an extremely dangerous thing to do, as it exposes the entirety of the database, including possibly sensitive information. Administrators and systems engineers are advised to instead use database-specific tools to make server transitions more securely.

Dumps the Traffic Ops database as an SQL script that should recreate its schema and contents exactly.

**Implementation Detail**

The script is output using the `pg_dump(1)` utility, and is thus compatible for use with the `pg_restore(1)` utility.

**GET**

Fetches the database dump.

- **Auth. Required** Yes
- **Roles Required** “admin”
- **Response Type** undefined - outputs an SQL script, not JSON

**Request Structure**

No parameters available

```
# 238: Request Example

GET /api/1.4/dbdump HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

**Response Structure**

```
# 239: 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/sql
Content-Disposition: attachment
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512:
X-Server-Name: traffic_ops_golang/
Date: Mon, 09 Sep 2019 21:08:28 GMT
Transfer-Encoding: chunked

--- Actual text omitted - it's huge

deliveryservice_server/{{DSID}}/{{serverID}}

DELETE

Removes a cache server from a Delivery Service.

Auth. Required  Yes

Roles Required  “admin” or “operations”

Response Type  undefined

Request Structure

Table 138: 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

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.
# 240: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: rGD2sOMHYF0sgalzuTtytyLHCukkc3ZwQRKvZ/HuPzObOP4WztKTOVXB4uhs3iJqBg9zRB2TucMxONHN+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

New in version 1.2.

**GET**

Retrieves time-aggregated statistics on a specific *Delivery Service*.

**Auth. Required** Yes

**Roles Required** None

Changed in version ATCv4: Prior to ATC version 4, this endpoint had more convoluted access requirements. The required *Roles* for access were “operations” or “admin” - *unless* the *Delivery Service* requested was “assigned” to the requesting user. In these older versions, Traffic Ops never checked the access to the *Delivery Service’s* stats from a *Tenant*-based perspective (which it now does) but instead used the legacy concept of “assigning” a *Delivery Service* to a user. It no longer considers such assignments whatsoever.

**Response Type** Object

---

1 This endpoint respects *Tenancy*, and users whose *Tenant* does not have access to a *Delivery Service* will be unable to view the statistics of said *Delivery Service*.
## Request Structure

Table 139: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>deliveryService</td>
<td>yes²</td>
<td>Either the xml_id of a Delivery Service for which statistics will be aggregated or the integral, unique identifier of said Delivery Service</td>
</tr>
<tr>
<td>deliveryServiceName</td>
<td>yes²</td>
<td>The xml_id of the Delivery Service for which statistics will be aggregated</td>
</tr>
<tr>
<td>endDate</td>
<td>yes</td>
<td>The date and time until which statistics shall be aggregated in RFC 3339 format (with or without sub-second precision), the number of nanoseconds since the Unix Epoch, or in the same, proprietary format as the lastUpdated fields prevalent throughout the Traffic Ops API</td>
</tr>
<tr>
<td>exclude</td>
<td>no</td>
<td>Either “series” to omit the data series from the result, or “summary” to omit the summary data from the result - directly corresponds to fields in the Response Structure</td>
</tr>
<tr>
<td>interval</td>
<td>no</td>
<td>Specifies the interval within which data will be “bucketed”; e.g. when requesting data from 2019-07-25T00:00:00Z to 2019-07-25T23:59:59Z with an interval of “1m”, the resulting data series (assuming it is not excluded) should contain $24\text{ hours} \times 60 \text{ minutes} \times 1 \text{ day} \times 1 \text{ minute} \times 1440 \text{data points} = 1440 \text{data points}$. The allowed values for this parameter are valid InfluxQL duration literal strings matching $^d+[mhdw]$ $^d+$</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>A natural number indicating the maximum amount of data points should be returned in the series object</td>
</tr>
<tr>
<td>metricType</td>
<td>yes</td>
<td>The metric type being reported - one of: kbps, The total traffic rate, out_bytes, The total number of bytes sent out to clients through the Delivery Service, status_4xx, The amount of requests that were serviced with 400-499 HTTP status codes, status_5xx, The amount of requests that were serviced with 500-599 HTTP status codes, tps_total, The total traffic rate in transactions per second served by the Delivery Service, tps_2xx, The total traffic rate in transactions per second serviced with 200-299 HTTP status codes, tps_3xx, The total traffic rate in transactions per second serviced with 300-399 HTTP status codes, tps_4xx, The total traffic rate in transactions per second serviced with 400-499 HTTP status codes, tps_5xx, The total traffic rate in transactions per second serviced with 500-599 HTTP status codes</td>
</tr>
</tbody>
</table>
New in version ATCv4: The `deliveryService` query parameter was added to this endpoint in all API versions in version 4 of ATC.

# 241: Request Example

```plaintext
GET /api/1.3/deliveryservice_stats?deliveryServiceName=demo1&
  startDate=2019-07-22T17:55:00Z&endDate=2019-07-22T17:56:00.000Z&
  metricType=tps_total HTTP/1.1
User-Agent: python-requests/2.20.1
Accept-Encoding: gzip, deflate
Accept: application/json;timestamp=unix, application/json;
  timestamp=rfc;q=0.9, application/json;q=0.8, */*;q=0.7
Connection: keep-alive
Cookie: mojolicious=...
```

## Content Format

It's important to note in Request Example the use of a complex “Accept” header. This endpoint accepts two special media types in the “Accept” header that instruct it on how to format the timestamps associated with the returned data. Specifically, Traffic Ops will recognize the special, optional, non-standard parameter of `application/json: timestamp`. The values of this parameter are restricted to one of

- **rfc**: Returned timestamps will be formatted according to RFC 3339 (no sub-second precision).
- **unix**: Returned timestamps will be formatted as the number of nanoseconds since the Unix Epoch (midnight on January 1st 1970 UTC).

## Implementation Detail

The endpoint passes back nanoseconds, specifically, because that is the form used both by InfluxDB, which is used to store the data being served, and Go’s standard library. Clients may need to convert the value to match their own standard libraries - e.g. the `Date()` class in Javascript expects milliseconds.

The default behavior - when only e.g. `application/json` or `*/*` is given - is to use RFC 3339 formatting. It will, however, respect quality parameters. It is suggested that clients request timestamps they can handle specifically, rather than relying on this default behavior, as it **is subject to change** and is in fact expected to invert in the next major release as string-based time formats become deprecated.

**See also:**

For more information on the “Accept” HTTP header, consult its dedicated page on MDN.

---

2 Either `deliveryServiceName` or `deliveryService` must be present, but if both are `deliveryServiceName` will be used and `deliveryService` will be ignored.
Response Structure

**series** An object containing the actual data series and information necessary for working with it.

  - **columns** This is an array of names of the columns of the data contained in the “values” array - should always be ["time", "sum_count"]
  - **count** The number of data points contained in the “values” array
  - **name** The name of the data set. Should always match `metric.ds.1min` where `metric` is the requested `metricType`
  - **values** The actual array of data points. Each represents a length of time specified by the `interval` query parameter
    - **time** The time at which the measurement was taken. This corresponds to the `beginning` of the interval. This time comes in the format of either an RFC 3339-formatted string, or a number containing the number of nanoseconds since the Unix Epoch depending on the “Accept” header sent by the client, according to the rules outlined in Content Format.
    - **value** The value of the requested `metricType` at the time given by `time`. This will always be a floating point number, unless no data is available for the data interval, in which case it will be `null`
  - **source** A legacy field meant only for plugins that override this endpoint to name themselves. Should always be “TrafficStats”.
    - Deprecated since version 1.4: As this has no known purpose, developers are advised it will be removed in the future.
  - **summary** An object containing summary statistics describing the data series
    - **average** The arithmetic mean of the data’s values
    - **count** The number of measurements taken within the requested timespan. This is, in general, **not** the same as the `count` field of the `series` object, as it reflects the number of underlying, un-“bucketed” data points, and is therefore dependent on the implementation of Traffic Stats.
    - **fifthPercentile** Data points with values less than or equal to this number constitute the “bottom” 5% of the data set
max  The maximum value that can be found in the requested data set

min  The minimum value that can be found in the requested data set

ninetyEighthPercentile  Data points with values greater than or equal to this number constitute the “top” 2% of the data set

ninetyFifthPercentile  Data points with values greater than or equal to this number constitute the “top” 5% of the data set

totalBytes  When the metricType requested is kbps, this will contain the total number of bytes transferred by the Delivery Service within the requested time window. Note that fractional amounts are possible, as the data transfer rate will almost certainly not be cleanly divided by the requested time range.

totalTransactions  When the metricType requested is not kbps, this will contain the total number of transactions completed by the Delivery Service within the requested time window. Note that fractional amounts are possible, as the transaction rate will almost certainly not be cleanly divided by the requested time range.

version  A legacy field that seems to have been meant to indicate the API version used. Will always be “1.2”

Deprecated since version 1.4: As this has no known purpose, developers are advised it will be removed in the future.

# 242: 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-Encoding: gzip
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: zXJGjcYuu6HxWINVp8HA1gL31J3ukry5wCsTDNxxtP/\waC6rSD8h10KJ9jEAtRzj9owOSVPvKaA/2bRu/QeuCpQ==
X-Server-Name: traffic_ops_golang/
Date: Mon, 22 Jul 2019 17:57:14 GMT
Transfer-Encoding: chunked

{ "response": {

(continues on next page)
"series": {
    "columns": [
        "time",
        "sum_count"
    ],
    "count": 2,
    "name": "tps_total.ds.1min",
    "tags": {
        "cachegroup": "total"
    },
    "values": [
        [1563818100000000000, 0],
        [1563818160000000000, 0]
    ]
},
"source": "TrafficStats",
"summary": {
    "average": 0,
    "count": 2,
    "fifthPercentile": 0,
    "max": 0,
    "min": 0,
    "ninetyEighthPercentile": 0,
    "ninetyFifthPercentile": 0,
    "totalBytes": null,
    "totalTransactions": 0
},
"version": "1.2"
}

deliveryservice_user

Deprecated since version ATCv4.

**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)

**# 243: 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 Services* assigned to the user identified by `userId`

**replace**  If `true`, any and all existing, conflicting *Delivery Service* assignments were overwritten by this assignment operation

**# 244: 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
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
```

(continues on next page)
Vary: Accept-Encoding
Whole-Content-Sha512:
Uw1+924m6Ye3NraFP+RBp1dkhcNTTDyXHZbzRaYV95p9tP56261gckeKSrl0QIkNXjXcCsDN5Dmum7
Content-Length: 127

```json
{
  "alerts": [
    {
      "level": "success",
      "text": "Delivery service assignments complete."
    },
    {
      "level": "warning",
      "text": "This endpoint and its functionality is deprecated, and will be removed in the future"
    }
  ],
  "response": {
    "userId": 5,
    "deliveryServices": [
      1
    ]
  }
}
```

deliveryservice_user/{{dsID}}/{{userID}}

Deprecated since version ATCv4.

**DELETE**

Removes a *Delivery Service* from a user.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** undefined

**Request Structure**

**Table 140: 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 <em>Delivery Service</em> which should no longer be assigned to the user identified by <code>userID</code></td>
</tr>
<tr>
<td>userId</td>
<td>An integral, unique identifier for the user to whom the <em>Delivery Service</em> identified by <code>dsID</code> should no longer be assigned</td>
</tr>
</tbody>
</table>
Response Structure

# 245: 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: Wed, 14 Nov 2018 21:40:06 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: /eNE3lhFABGukcczxjXOYiwmfVTUUKII9RRuZil4AbF65BlhHdXZ5lAVEi4Hc65+jNajBgI9{jmsgO
Content-Length: 100

{ "alerts": [
    {
      "level": "success",
      "text": "User [ test ] unlinked from deliveryservice [ 1 | demo1 ]."
    },
    {
      "level": "warning",
      "text": "This endpoint and its functionality is deprecated, and will be removed in the future"
    }
]}
```

deliveryservices

GET

Retrieves Delivery Services

Auth. Required Yes

Roles Required None

Response Type Array

Only those Delivery Services assigned to Tenants that are the requesting user’s Tenant or children thereof will appear in the output of a GET request, and the same constraints are placed on the allowed values of the tenantId field of a POST request to create a new Delivery Service
Request Structure

Table 141: 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>Show only the Delivery Services that have Logs Enabled set or not based on this boolean</td>
</tr>
<tr>
<td>profile</td>
<td>no</td>
<td>Return only Delivery Services using the Profile that has this ID</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>
<tr>
<td>orderBy</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n^th page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

Response Structure

active  A boolean that defines Active.

anonymousBlockingEnabled  A boolean that defines Anonymous Blocking

cacheurl  A Cache URL Expression

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 DNS TTL - 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  A Check Path

consistentHashRegex  A Consistent Hashing Regular Expression

New in version 1.4.
**consistentHashQueryParams**: An array of *Consistent Hashing Query Parameters*

- New in version 1.4.

**deepCachingType**: The *Deep Caching* setting for this *Delivery Service*

- New in version 1.3.

**displayName**: The *Display Name*

**dnsBypassCname**: A *DNS Bypass CNAME*

**dnsBypassIp**: A *DNS Bypass IP*

**dnsBypassIp6**: A *DNS Bypass IPv6*

**dnsBypassTtl**: The *DNS Bypass TTL*

**dscp**: A *DSCP* to be used within the *Delivery Service*

**ecsEnabled**: A boolean that defines the *EDNS0 Client Subnet Enabled* setting on this *Delivery Service*

- New in version 1.5.

**edgeHeaderRewrite**: A set of *Edge Header Rewrite Rules*

**exampleURLs**: An array of *Example URLs*

**fqPacingRate**: The *Fair-Queuing Pacing Rate Bps*

- New in version 1.3.

**geoLimit**: An integer that defines the *Geo Limit*

**geoLimitCountries**: A string containing a comma-separated list defining the *Geo Limit Countries*

**geoLimitRedirectUrl**: A *Geo Limit Redirect URL*

**geoProvider**: The *Geolocation Provider*

**globalMaxMbps**: The *Global Max Mbps*

**globalMaxTps**: The *Global Max TPS*

**httpBypassFqdn**: A *HTTP Bypass FQDN*

**id**: An integral, unique identifier for this *Delivery Service*

**infoUrl**: An *Info URL*

**initialDispersion**: The *Initial Dispersion*

**ipv6RoutingEnabled**: A boolean that defines the *IPv6 Routing Enabled* setting on this *Delivery Service*

**lastUpdated**: The date and time at which this *Delivery Service* was last updated, in RFC 3339 format

**logsEnabled**: A boolean that defines the *Logs Enabled* setting on this *Delivery Service*
longDesc  The *Long Description* of this *Delivery Service*

longDesc1  The *Long Description 2* of this *Delivery Service*

longDesc2  The *Long Description 3* of this *Delivery Service*

matchList  The *Delivery Service’s Match List*

  pattern  A regular expression - the use of this pattern is dependent on the *type* field (backslashes are escaped)

  setNumber  An integer that provides explicit ordering of *Match List* items - this is used as a priority ranking by Traffic Router, and is not guaranteed to correspond to the ordering of items in the array.

  type  The type of match performed using *pattern*.

maxDnsAnswers  The *Max DNS Answers* allowed for this *Delivery Service*

maxOriginConnections  The *Max Origin Connections*

  New in version 1.4.

midHeaderRewrite  A set of *Mid Header Rewrite Rules*

missLat  The *Geo Miss Default Latitude* used by this *Delivery Service*

missLong  The *Geo Miss Default Longitude* used by this *Delivery Service*

multiSiteOrigin  A boolean that defines the use of *Use Multi-Site Origin Feature* by this *Delivery Service*

orgServerFqdn  The *Origin Server Base URL*

originShield  A *Origin Shield* string

profileDescription  The *Description* of the *Profile* with which this *Delivery Service* is associated

profileId  The *ID* of the *Profile* with which this *Delivery Service* is associated

profileName  The *Name* of the *Profile* with which this *Delivery Service* is associated

protocol  An integral, unique identifier that corresponds to the *Protocol* used by this *Delivery Service*

qstringIgnore  An integral, unique identifier that corresponds to the *Query String Handling* setting on this *Delivery Service*

rangeRequestHandling  An integral, unique identifier that corresponds to the *Range Request Handling* setting on this *Delivery Service*

regexRemap  A *Regex Remap Expression*

regionalGeoBlocking  A boolean defining the *Regional Geoblocking* setting on this *Delivery Service*

remapText  *Raw Remap Text*
signed  true if and only if signingAlgorithm is not null, false otherwise

signingAlgorithm Either a Signing Algorithm or null to indicate URL/URI signing is not implemented on this Delivery Service

New in version 1.3.

sslKeyVersion This integer indicates the SSL Key Version

tenantId The integral, unique identifier of the Tenant who owns this Delivery Service

New in version 1.3.

trRequestHeaders If defined, this defines the Traffic Router Log Request Headers used by Traffic Router for this Delivery Service

New in version 1.3.

trResponseHeaders If defined, this defines the Traffic Router Additional Response Headers used by Traffic Router for this Delivery Service

New in version 1.3.

type The Type of this Delivery Service

typeId The integral, unique identifier of the Type of this Delivery Service

xmlId This Delivery Service’s xml_id

# 246: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: mCLMjvACRKHNGP/5OSx4javkOtxxyiDdQzsV78IamUhVmvvKyKaCeOKRmpsG69w+nhh30kPZ6e9MMeJpcJSKcA==
X-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",

(continues on next page)
"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": "2019-05-15 14:32:05+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": 2,
"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.demol.mycdn.ciab.test",
  "https://video.demol.mycdn.ciab.test"
],
"deepCachingType": "NEVER",
"fqPacingRate": null,
"signingAlgorithm": null,
"tenant": "root",
"trResponseHeaders": null,
"trRequestHeaders": null,
"consistentHashRegex": null,
"consistentHashQueryParams": [  
  "abc",
  "pdq",
  "xxx",
  "zyx"
],
"maxOriginConnections": 0,
"ecsEnabled": false
}}

**POST**

Allows users to create *Delivery Service*.

**Auth. Required**  Yes

**Roles Required**  “admin” or “operations”

**Response Type**  Array

**Request Structure**

- **active**  A boolean that defines *Active*.
- **anonymousBlockingEnabled**  A boolean that defines *Anonymous Blocking*
- **cacheurl**  A Cache URL Expression

  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 DNS TTL - named “ccrDnsTtl” for legacy reasons
cdnId  The integral, unique identifier of the CDN to which the Delivery Service belongs

checkPath  A Check Path

consistentHashRegex  A Consistent Hashing Regular Expression

consistentHashQueryParams  An array of Consistent Hashing Query Parameters

deepCachingType  The Deep Caching setting for this Delivery Service

displayName  The Display Name

dnsBypassCname  A DNS Bypass CNAME

dnsBypassIp  A DNS Bypass IP

dnsBypassIp6  A DNS Bypass IPv6

dnsBypassTtl  The DNS Bypass TTL

dscp  A DSCP to be used within the Delivery Service

cesEnabled  A boolean that defines the EDNS0 Client Subnet Enabled setting on this Delivery Service

edgeHeaderRewrite  A set of Edge Header Rewrite Rules

fqPacingRate  The Fair-Queuing Pacing Rate Bps

geoLimit  An integer that defines the Geo Limit

geoLimitCountries  A string containing a comma-separated list defining the Geo Limit Countries

geoLimitRedirectUrl  A Geo Limit Redirect URL

geoProvider  The Geolocation Provider

globalMaxMbps  The Global Max Mbps

globalMaxTps  The Global Max TPS

httpBypassFqdn  A HTTP Bypass FQDN

infoUrl  An Info URL

initialDispersion  The Initial Dispersion

These fields must be defined if and only if geoLimit is non-zero
ipv6RoutingEnabled A boolean that defines the IPv6 Routing Enabled setting on this Delivery Service

logsEnabled A boolean that defines the Logs Enabled setting on this Delivery Service

longDesc The Long Description of this Delivery Service

longDesc1 An optional field containing the Long Description 2 of this Delivery Service

longDesc2 An optional field containing the Long Description 3 of this Delivery Service

maxDnsAnswers The Max DNS Answers allowed for this Delivery Service

maxOriginConnections The Max Origin Connections

   New in version 1.4.

midHeaderRewrite A set of Mid Header Rewrite Rules

missLat The Geo Miss Default Latitude used by this Delivery Service

missLong The Geo Miss Default Longitude used by this Delivery Service

multiSiteOrigin A boolean that defines the use of Use Multi-Site Origin Feature by this Delivery Service

orgServerFqdn The Origin Server Base URL

originShield A Origin Shield string

profileId An optional ID of a Profile with which this Delivery Service shall be associated

protocol An integral, unique identifier that corresponds to the Protocol used by this Delivery Service

queryStringIgnore An integral, unique identifier that corresponds to the Query String Handling setting on this Delivery Service

rangeRequestHandling An integral, unique identifier that corresponds to the Range Request Handling setting on this Delivery Service

regexRemap A Regex Remap Expression

regionalGeoBlocking A boolean defining the Regional Geoblocking setting on this Delivery Service

remapText Raw Remap Text

signed true if and only if signingAlgorithm is not null, false otherwise

signingAlgorithm Either a Signing Algorithm or null to indicate URL/URI signing is not implemented on this Delivery Service

   New in version 1.3.
sslKeyVersion  This integer indicates the SSL Key Version

tenantId  The integral, unique identifier of the Tenant who owns this Delivery Service

New in version 1.3.

trRequestHeaders  If defined, this defines the Traffic Router Log Request Headers used by Traffic Router for this Delivery Service

New in version 1.3.

trResponseHeaders  If defined, this defines the Traffic Router Additional Response Headers used by Traffic Router for this Delivery Service

New in version 1.3.

type  The Type of this Delivery Service

typeId  The integral, unique identifier of the Type of this Delivery Service

xmlId  This Delivery Service’s xml_id

# 247: 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,
    "deepCachingType": "NEVER",
    "displayName": "test",
    "dscp": 0,
    "ecsEnabled": true,
    "geoLimit": 0,
    "geoProvider": 0,
    "initialDispersion": 1,
    "ipv6RoutingEnabled": false,
    "logsEnabled": true,
    "longDesc": "A Delivery Service created expressly for API documentation examples",
    "missLat": 0,
    "missLong": 0,
    "maxOriginConnections": 0,
    "multiSiteOrigin": false,
    "orgServerFqdn": "http://origin.infra.ciab.test",
    "protocol": 0,
}
```

(continues on next page)
"qstringIgnore": 0,
"rangeRequestHandling": 0,
"regionalGeoBlocking": false,
"routingName": "test",
"signed": false,
"tenant": "root",
"tenantId": 1,
"typeId": 1,
"xmlId": "test"
}

Response Structure

**active**  A boolean that defines *Active*.

**anonymousBlockingEnabled**  A boolean that defines *Anonymous Blocking*

**cacheurl**  A *Cache URL Expression*

   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 *DNS TTL* - 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**  A *Check Path*

**consistentHashRegex**  A *Consistent Hashing Regular Expression*

   New in version 1.4.

**consistentHashQueryParams**  An array of *Consistent Hashing Query Parameters*

   New in version 1.4.

**deepCachingType**  The *Deep Caching* setting for this *Delivery Service*

   New in version 1.3.

**displayName**  The *Display Name*

**dnsBypassCname**  A *DNS Bypass CNAME*

**dnsBypassIp**  A *DNS Bypass IP*

**dnsBypassIp6**  A *DNS Bypass IPv6*

**dnsBypassTtl**  The *DNS Bypass TTL*

**dscp**  A *DSCP* to be used within the *Delivery Service*
**ecsEnabled**  A boolean that defines the *EDNS0 Client Subnet Enabled* setting on this *Delivery Service*

New in version 1.4.

**edgeHeaderRewrite**  A set of *Edge Header Rewrite Rules*

**exampleURLs**  An array of *Example URLs*

**fqPacingRate**  The *Fair-Queuing Pacing Rate Bps* 

New in version 1.3.

**geoLimit**  An integer that defines the *Geo Limit*

**geoLimitCountries**  A string containing a comma-separated list defining the *Geo Limit Countries*

**geoLimitRedirectUrl**  A *Geo Limit Redirect URL*

**geoProvider**  The *Geolocation Provider*

**globalMaxMbps**  The *Global Max Mbps*

**globalMaxTps**  The *Global Max TPS*

**httpBypassFqdn**  A *HTTP Bypass FQDN*

**id**  An integral, unique identifier for this *Delivery Service*

**infoUrl**  An *Info URL*

**initialDispersion**  The *Initial Dispersion*

**ipv6RoutingEnabled**  A boolean that defines the *IPv6 Routing Enabled* setting on this *Delivery Service*

**lastUpdated**  The date and time at which this *Delivery Service* was last updated, in *RFC 3339* format

**logsEnabled**  A boolean that defines the *Logs Enabled* setting on this *Delivery Service*

**longDesc**  The *Long Description* of this *Delivery Service*

**longDesc1**  The *Long Description 2* of this *Delivery Service*

**longDesc2**  The *Long Description 3* of this *Delivery Service*

**matchList**  The *Delivery Service*s *Match List*

  **pattern**  A regular expression - the use of this pattern is dependent on the *type* field (backslashes are escaped)

  **setNumber**  An integer that provides explicit ordering of *Match List* items - this is used as a priority ranking by Traffic Router, and is not guaranteed to correspond to the ordering of items in the array.

  **type**  The type of match performed using *pattern*. 
maxDnsAnswers  The Max DNS Answers allowed for this Delivery Service

maxOriginConnections  The Max Origin Connections

New in version 1.4.

midHeaderRewrite  A set of Mid Header Rewrite Rules

missLat  The Geo Miss Default Latitude used by this Delivery Service

missLong  The Geo Miss Default Longitude used by this Delivery Service

multiSiteOrigin  A boolean that defines the use of Use Multi-Site Origin Feature by this Delivery Service

orgServerFqdn  The Origin Server Base URL

originShield  A Origin Shield string

profileDescription  The Description of the Profile with which this Delivery Service is associated

profileId  The ID of the Profile with which this Delivery Service is associated

profileName  The Name of the Profile with which this Delivery Service is associated

protocol  An integral, unique identifier that corresponds to the Protocol used by this Delivery Service

qstringIgnore  An integral, unique identifier that corresponds to the Query String Handling setting on this Delivery Service

rangeRequestHandling  An integral, unique identifier that corresponds to the Range Request Handling setting on this Delivery Service

regexRemap  A Regex Remap Expression

regionalGeoBlocking  A boolean defining the Regional Geoblocking setting on this Delivery Service

remapText  Raw Remap Text

signed  true if and only if signingAlgorithm is not null, false otherwise

signingAlgorithm  Either a Signing Algorithm or null to indicate URL/URI signing is not implemented on this Delivery Service

New in version 1.3.

sslKeyVersion  This integer indicates the SSL Key Version

tenantId  The integral, unique identifier of the Tenant who owns this Delivery Service

New in version 1.3.

trRequestHeaders  If defined, this defines the Traffic Router Log Request Headers used by Traffic Router for this Delivery Service
New in version 1.3.

**trResponseHeaders** If defined, this defines the *Traffic Router Additional Response Headers* used by Traffic Router for this *Delivery Service*

New in version 1.3.

**type** The *Type* of this *Delivery Service*

**typeId** The integral, unique identifier of the *Type* of this *Delivery Service*

**xmlId** This *Delivery Service*’s `xml_id`

---

### # 248: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Accept, Set-Cookie, Content-Type, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: SVveQShGwfPv8N5APUkLozrTUVA+z8wuFLsS1Cr1/vvFJ0VQOMUCTg1NbhAuQ795MJmuuAaAwR8dSOQ==
X-Server-Name: traffic_ops_golang/
Date: Mon, 19 Nov 2018 19:45:49 GMT
Content-Length: 1404

{  "alerts": [    {        "text": "Delivery service creation was successful.",        "level": "success"    }  ],  "response": [    {        "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,    }  ]
```

(continues on next page)
"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 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"
]
**deliveryservices/dnsseckeys/generate**

**POST**

Generates ZSK and KSK keypairs for a CDN and all associated Delivery Services.

**Auth. Required** Yes

**Roles Required** “admin”

**Response Type** Object (string)

**Request Structure**

- `effectiveDate` UNIX epoch start date for the signing keys
  
  New in version 1.2.

- `key` Name of the CDN

- `kskExpirationDays` Expiration (in days) for the KSKs (Key-Signing Keys)

- `name` Domain name used by the CDN

- `ttl` Time for which the keypairs shall remain valid

- `zskExpirationDays` Expiration (in days) for the ZSKs (Zone-Signing Keys)

**Response Structure**

# 249: Response Example

```json
{
  "response": "Successfully created dnssec keys for cdn1"
}
```

**deliveryservices/hostname/{{name}}/sslkeys**
GET

Auth. Required  Yes
Roles Required “admin”
Response Type  Object

Request Structure

Table 142: 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 143: 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>

# 250: 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\(^1\)

**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

---

\(^1\) 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.
**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 string that is 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 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

expiration The expiration date of the certificate for the Delivery Service in RFC 3339 format

New in version 1.5.

# 251: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: eXZMeGFYSJgjg/rC1JtHpqMHNEvxx2Gw2kCXS15mOFF+oU74UXmRK1/2KF+kLyNhGTSceZ07m/
qqpwDLgv1GOEg==
```


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 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}}

GET

Caution: It’s often much better to the `id` query parameter of a GET request to `delivery-services` instead.

Retrieves a specific Delivery Service

Auth. Required Yes

Roles Required None¹

Response Type Array

¹ Only those Delivery Services assigned to Tenants that are the requesting user’s Tenant or children thereof will appear in the output of a GET request, and the same constraints are placed on the allowed values of the `tenantId` field of a PUT request to update a new Delivery Service. Furthermore, the only Delivery Services a user may delete are those assigned to a Tenant that is either the same Tenant as the user’s Tenant, or a descendant thereof.
Request Structure

Table 144: 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>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 with this ID</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>
<tr>
<td>orderBy</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

Table 145: 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 to be retrieved</td>
</tr>
</tbody>
</table>

Response Structure

- **active** A boolean that defines Active.
- **anonymousBlockingEnabled** A boolean that defines Anonymous Blocking
- **cacheurl** A Cache URL Expression
  
  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 DNS TTL - 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  A Check Path
consistentHashRegex  A Consistent Hashing Regular Expression
   New in version 1.4.
consistentHashQueryParams  An array of Consistent Hashing Query Parameters
   New in version 1.4.
depthCachingType  The Deep Caching setting for this Delivery Service
   New in version 1.3.
displayName  The Display Name
dnsBypassCname  A DNS Bypass CNAME
dnsBypassIp  A DNS Bypass IP
dnsBypassIp6  A DNS Bypass IPv6
dnsBypassTtl  The DNS Bypass TTL
dscp  A DSCP to be used within the Delivery Service
ecsEnabled  A boolean that defines the EDNS0 Client Subnet Enabled setting on this Delivery Service
   New in version 1.4.
edgeHeaderRewrite  A set of Edge Header Rewrite Rules
eexampleURLs  An array of Example URLs
fqPacingRate  The Fair-Queuing Pacing Rate Bps
   New in version 1.3.
geoLimit  An integer that defines the Geo Limit
geoLimitCountries  A string containing a comma-separated list defining the Geo Limit Countries
geoLimitRedirectUrl  A Geo Limit Redirect URL
geoProvider  The Geolocation Provider
globalMaxMbps  The Global Max Mbps
globalMaxTps  The Global Max TPS
httpBypassFqdn  A HTTP Bypass FQDN
id  An integral, unique identifier for this Delivery Service
infoUrl  An Info URL
initialDispersion  The Initial Dispersion
ipv6RoutingEnabled  A boolean that defines the IPv6 Routing Enabled setting on this Delivery Service
lastUpdated  The date and time at which this Delivery Service was last updated, in RFC 3339 format

logsEnabled  A boolean that defines the Logs Enabled setting on this Delivery Service

longDesc  The Long Description of this Delivery Service

longDesc1  The Long Description 2 of this Delivery Service

longDesc2  The Long Description 3 of this Delivery Service

matchList  The Delivery Service’s Match List

  pattern  A regular expression - the use of this pattern is dependent on the type field (backslashes are escaped)

  setNumber  An integer that provides explicit ordering of Match List items - this is used as a priority ranking by Traffic Router, and is not guaranteed to correspond to the ordering of items in the array.

  type  The type of match performed using pattern.

maxDnsAnswers  The Max DNS Answers allowed for this Delivery Service

maxOriginConnections  The Max Origin Connections

  New in version 1.4.

midHeaderRewrite  A set of Mid Header Rewrite Rules

missLat  The Geo Miss Default Latitude used by this Delivery Service

missLong  The Geo Miss Default Longitude used by this Delivery Service

multiSiteOrigin  A boolean that defines the use of Use Multi-Site Origin Feature by this Delivery Service

orgServerFqdn  The Origin Server Base URL

originShield  A Origin Shield string

profileDescription  The Description of the Profile with which this Delivery Service is associated

profileId  The ID of the Profile with which this Delivery Service is associated

profileName  The Name of the Profile with which this Delivery Service is associated

protocol  An integral, unique identifier that corresponds to the Protocol used by this Delivery Service

qstringIgnore  An integral, unique identifier that corresponds to the Query String Handling setting on this Delivery Service

rangeRequestHandling  An integral, unique identifier that corresponds to the Range Request Handling setting on this Delivery Service
regexRemap  A *Regex Remap Expression*

regionalGeoBlocking  A boolean defining the *Regional Geoblocking* setting on this *Delivery Service*

remapText  *Raw Remap Text*

signed  True if and only if `signingAlgorithm` is not null, false otherwise

signingAlgorithm  Either a *Signing Algorithm* or null to indicate URL/URI signing is not implemented on this *Delivery Service*

New in version 1.3.

sslKeyVersion  This integer indicates the *SSL Key Version*

tenantId  The integral, unique identifier of the *Tenant* who owns this *Delivery Service*

New in version 1.3.

trRequestHeaders  If defined, this defines the *Traffic Router Log Request Headers* used by Traffic Router for this *Delivery Service*

New in version 1.3.

trResponseHeaders  If defined, this defines the *Traffic Router Additional Response Headers* used by Traffic Router for this *Delivery Service*

New in version 1.3.

type  The *Type* of this *Delivery Service*

typeId  The integral, unique identifier of the *Type* of this *Delivery Service*

xmlId  This *Delivery Service’s xml_id*
{ "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": "2019-06-10 13:05:19+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": 2,
}]}
"qstringIgnore": 0,
"rangeRequestHandling": 0,
"regexRemap": null,
"regionalGeoBlocking": false,
"remapText": null,
"routingName": "video",
"signed": false,
"sslKeyVersion": 1,
"tenantId": 1,
"type": "HTTP",
"tenId": 1,
"xmlId": "demo1",
"exampleURLs": [
  "http://video.demo1.mycdn.ciab.test",
  "https://video.demo1.mycdn.ciab.test"
],
"deepCachingType": "NEVER",
"fqPacingRate": null,
"signingAlgorithm": null,
"tenant": "root",
"trResponseHeaders": null,
"trRequestHeaders": null,
"consistentHashRegex": null,
"consistentHashQueryParams": [
  "abc",
  "pdq",
  "xxx",
  "zyx"
],
"maxOriginConnections": 0,
"ecsEnabled": false
}]

PUT

Allows users to edit an existing Delivery Service.

Auth. Required  Yes

Roles Required  “admin” or “operations”1

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**  A boolean that defines *Active*.

**anonymousBlockingEnabled**  A boolean that defines *Anonymous Blocking*.

**cacheurl**  A *Cache URL Expression*

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 *DNS TTL* - named “ccrDnsTtl” for legacy reasons.

**cdnId**  The integral, unique identifier of the *CDN* to which the *Delivery Service* belongs.

**checkPath**  A *Check Path*.

**consistentHashRegex**  A *Consistent Hashing Regular Expression*

New in version 1.4.

**consistentHashQueryParams**  An array of *Consistent Hashing Query Parameters*

New in version 1.4.

**deepCachingType**  The *Deep Caching* setting for this *Delivery Service*

New in version 1.3.

**displayName**  The *Display Name*.

**dnsBypassCname**  A *DNS Bypass CNAME*.

**dnsBypassIp**  A *DNS Bypass IP*.

**dnsBypassIp6**  A *DNS Bypass IPv6*.

**dnsBypassTtl**  The *DNS Bypass TTL*.

**dscp**  A *DSCP* to be used within the *Delivery Service*.

**ecsEnabled**  A boolean that defines the *EDNS0 Client Subnet Enabled* setting on this *Delivery Service*.

New in version 1.4.

**edgeHeaderRewrite**  A set of *Edge Header Rewrite Rules*.

**fqPacingRate**  The *Fair-Queuing Pacing Rate Bps*.

New in version 1.3.

**geoLimit**  An integer that defines the *Geo Limit*.

**geoLimitCountries**  A string containing a comma-separated list defining the *Geo Limit Countries*.

**geoLimitRedirectUrl**  A *Geo Limit Redirect URL*.

---

1 These fields must be defined if and only if `geoLimit` is non-zero.
geoProvider  The Geolocation Provider

globalMaxMbps  The Global Max Mbps

globalMaxTps  The Global Max TPS

httpBypassFqdn  A HTTP Bypass FQDN

infoUrl  An Info URL

initialDispersion  The Initial Dispersion

ipv6RoutingEnabled  A boolean that defines the IPv6 Routing Enabled setting on this Delivery Service

logsEnabled  A boolean that defines the Logs Enabled setting on this Delivery Service

longDesc  The Long Description of this Delivery Service

longDesc1  An optional field containing the Long Description 2 of this Delivery Service

longDesc2  An optional field containing the Long Description 3 of this Delivery Service

maxDnsAnswers  The Max DNS Answers allowed for this Delivery Service

maxOriginConnections  The Max Origin Connections

  New in version 1.4.

midHeaderRewrite  A set of Mid Header Rewrite Rules

missLat  The Geo Miss Default Latitude used by this Delivery Service

missLong  The Geo Miss Default Longitude used by this Delivery Service

multiSiteOrigin  A boolean that defines the use of Use Multi-Site Origin Feature by this Delivery Service

orgServerFqdn  The Origin Server Base URL

originShield  A Origin Shield string

profileId  An optional ID of the Profile with which this Delivery Service will be associated

protocol  An integral, unique identifier that corresponds to the Protocol used by this Delivery Service

qstringIgnore  An integral, unique identifier that corresponds to the Query String Handling setting on this Delivery Service

rangeRequestHandling  An integral, unique identifier that corresponds to the Range Request Handling setting on this Delivery Service

regexRemap  A Regex Remap Expression
regionalGeoBlocking A boolean defining the Regional Geoblocking setting on this Delivery Service

remapText Raw Remap Text

routingName The Routing Name of this Delivery Service

signed true if and only if signingAlgorithm is not null, false otherwise

signingAlgorithm Either a Signing Algorithm or null to indicate URL/URI signing is not implemented on this Delivery Service

New in version 1.3.

sslKeyVersion This integer indicates the SSL Key Version

tenantId The integral, unique identifier of the Tenant who owns this Delivery Service

New in version 1.3.

trRequestHeaders If defined, this defines the Traffic Router Log Request Headers used by Traffic Router for this Delivery Service

New in version 1.3.

trResponseHeaders If defined, this defines the Traffic Router Additional Response Headers used by Traffic Router for this Delivery Service

New in version 1.3.

typeId The integral, unique identifier of the Type of this Delivery Service

xmlId This Delivery Service’s xml_id

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.

# 253: Request Example

```
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

{
   "active": true,
   "anonymousBlockingEnabled": false,
   "cdnId": 2,
   "tenantId": 4,
   "typeId": 5,
   "signed": true,
   "regionalGeoBlocking": true,
   "remapText": "Example Remap Text",
   "routingName": "My Delivery Service",
   "signed": true,
   "signingAlgorithm": "HMAC-SHA256",
   "sslKeyVersion": 1,
   "tenantId": 4
}
```
"cdnName": "CDN-in-a-Box",
"deepCachingType": "NEVER",
"displayName": "demo",
"dscp": 0,
"ecsEnabled": true,
"geoLimit": 0,
"geoProvider": 0,
"initialDispersion": 1,
"ipv6RoutingEnabled": false,
"lastUpdated": "2018-11-14 18:21:17+00",
"logsEnabled": true,
"longDesc": "A 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

# 254: 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: z4PhNX7vul3xVChq1m2AB9Yg5AULVxXcg/LspIdNs6c5H0E8XYXysP+DGNKHfuwvY7kxvUdBeoG1ODJ6+SfaPg==
X-Server-Name: traffic_ops_golang/
Date: Tue, 20 Nov 2018 14:12:25 GMT
Content-Length: 0
Content-Type: text/plain; charset=utf-8
```
DELETE

Deletes the target Delivery Service

Auth. Required Yes
Roles Required “admin” or “operations”1
Response Type undefined

Request Structure

Table 146: 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 to be deleted</td>
</tr>
</tbody>
</table>

# 255: 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

# 256: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: w9NlQpJJE15r6iYq/ffk8o5WfAXeUS5XR9yDhvKUgPO81yEo8YyftaSF0MPFseeOk60dk6kQo+MLYTDIAhhRwx==
X-Server-Name: traffic_ops_golang/
Date: Tue, 20 Nov 2018 14:56:37 GMT
Content-Length: 57

{ "alerts": [
  {
    "text": "ds was deleted."
  }
]
"level": "success"
}
]
}

deliveryservices/{{ID}}/capacity

See also:
Health Protocol

GET

Retrieves the usage percentages of a servers associated with a Delivery Service

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>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

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

1 Users will only be able to see capacity details for the Delivery Services their Tenant is allowed to see.
# 257: 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=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: ++dFR9V1c60CHGNwMjX6JSFEjHreXcL4QnhT03hiv04ByY379aLpL4OrOzX2bPgJgpR94+f6jZ0+iD2RS
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

**Roles Required** None

**Response Type** Object

---

1 Users will only be able to see Cache Group health details for the Delivery Services their Tenant is allowed to see.
Request Structure

Table 148: 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 <em>Cache Groups</em> will be displayed</td>
</tr>
</tbody>
</table>

Response Structure

```py
接入 groups An array of objects that represent the health of each *Cache Group* assigned to this *Delivery Service*

name A string that is 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*
```

# 258: Response Example

```
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
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=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: KpXViXeAgch58ueQgdyU8NuINBw1EUedE6Rv2ewcLUajJp6kowdbVypwW7XiSvAyHdtClIOuT3OhkR8G1Ev15
Content-Length: 115

{ "response": {
    "totalOffline": 0,
    "totalOnline": 1,
}}
```

(continues on next page)
"cachegroups": [
    {
        "offline": 0,
        "name": "CDN_in_a_Box_Edge",
        "online": 1
    }
]
}

deliveryservices/{{ID}}/regexes

GET

Retrieves routing regular expressions for a specific Delivery Service.

Auth. Required Yes

Roles Required None

Response Type Array

Request Structure

Table 149: 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>

# 259: 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=...
```

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

1 Users will only be able to view and create regular expressions for the Delivery Services their Tenant is allowed to see.
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

# 260: 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: fW9Fde4WRpp2ShRAC41P9s/PhU71LI/SEZhgYjGqfzhk45Wq0kpaWy76JvPfLpowY8eDTp8Y8TL5rNGEc+bM+A==
X-Server-Name: traffic_ops_golang/
Date: Tue, 27 Nov 2018 20:56:43 GMT
Content-Length: 100

{
  "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”!

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 the Delivery Service being inspected</td>
</tr>
<tr>
<td>pattern</td>
<td>The actual regular expression</td>
</tr>
</tbody>
</table>

5.1. Traffic Ops API
**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

### # 261: Request Example

```
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

### # 262: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: kS5dRzAhFKE7vfzHK7XIwpM0jztksk9MU+qtj5YU/1oxVHmqNbJ12Fe0OJJsZJCXbY1nBS04sCl95Sz5wed1Q==
```

(continues on next page)
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

Request Structure

Table 151: 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>

# 263: Request Example

GET /api/1.4/deliveryservices/1/regexes/1 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0

(continues on next page)

1 Users will only be able to view, delete and update regular expressions for the Delivery Services their Tenant is allowed 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

# 264: 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=Mon, 18 Nov 2019
  17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: fW9Fde4WRpp2ShRAC41P9s/PhU71LI/SEzHqYjGqfzhk45wq0kpaWy76JvPfLpowY8eDTp8Y8TL5rNGEc+bM+A==
X-Server-Name: traffic_ops_golang/
Date: Tue, 27 Nov 2018 21:08:34 GMT
Content-Length: 100

{
  "response": [
    {
      "id": 1,
      "type": 31,
      "typeName": "HOST_REGEXP",
      "setNumber": 0,
      "pattern": ".*\\.demo1\\..*"
    }
  ]
}
```

**PUT**

Updates a routing regular expression.

**Auth. Required**  Yes
Roles Required  “admin” or “operations”¹

Response Type  Object

Request Structure

Table 152: 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

# 265: Request Example

```plaintext
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

{
    "pattern": ".*\\.foo-bar\\..*",
    "type": 33,
    "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

5.1. Traffic Ops API
### typeName

The type of regular expression - determines that against which it will be evaluated

#### # 266: Response Example

<table>
<thead>
<tr>
<th>HTTP/1.1 200 OK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access-Control-Allow-Credentials: true</td>
</tr>
<tr>
<td>Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Cookie, Cookie</td>
</tr>
<tr>
<td>Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE</td>
</tr>
<tr>
<td>Access-Control-Allow-Origin: *</td>
</tr>
<tr>
<td>Content-Type: application/json</td>
</tr>
<tr>
<td>Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly</td>
</tr>
<tr>
<td>Whole-Content-Sha512: kS5dRzAhFKE7vzfzHK7XVIwpMOjztksk9MU+qtj5YU/loxBvqNfYj12Fb0LOs7ZJXbY1nBS04sCI95Sz5wed1Q==</td>
</tr>
<tr>
<td>X-Server-Name: traffic_ops_golang/</td>
</tr>
<tr>
<td>Date: Thu, 29 Nov 2018 17:54:58 GMT</td>
</tr>
<tr>
<td>Content-Length: 188</td>
</tr>
</tbody>
</table>

```json

{ "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”!

**Response Type**  undefined
Request Structure

Table 153: 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>

# 267: 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

# 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=/; Expires=Mon, 18 Nov 2019
            17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: 8oEa78x7f/o39LIS98W6G+UqE6cX/
                       1w4v3mMHvbAsliWHALuDYRz3V0tA6jzfGQKpB040m8qaVG+zwRrBVoCmQ==
X-Server-Name: traffic_ops_golang/
Date: Thu, 29 Nov 2018 18:44:00 GMT
Content-Length: 76

{ "alerts": [
    {
        "text": "deliveryservice_regex was deleted.",
        "level": "success"
    }
]}
```

deliveryservices/{{ID}}/routing

GET

Retrieves routing method statistics for a particular Delivery Service
Auth. Required  Yes
Roles Required  None
Response Type  Object

Request Structure

Table 154: 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>

# 269: 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=...
```

Response Structure

cz  The percent of requests to the Traffic Router for this Delivery Service that were satisfied by a Coverage Zone File

deepCz  The percent of requests to the Traffic Router for this Delivery Service that were satisfied by a Deep Coverage Zone File

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

---

1 Users will only be able to view routing details for the Delivery Services their Tenant is allowed to see.
**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 *Static DNS Entries*

### # 270: 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
Date: Fri, 30 Nov 2018 15:08:07 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: UgPziRC/5u4+Cfk29xm0EkEzjjJVu6cwBrFd/n3xH/0zmlkaXkqaaly4+B7dyE46vxFLYE0OdOcQchyn7JkoQog==
Content-Length: 132

{  "response": {  "cz": 79,  "deepCz": 0.50,  "dsr": 0,  "err": 0,  "fed": 0.25,  "geo": 20,  "miss": 0.25,  "regionalAlternate": 0,  "regionalDenied": 0,  "staticRoute": 0  }}
```

deliveryservices/{{ID}}/safe

**PUT**

Allows a user to edit metadata fields of a *Delivery Service*.

**Auth. Required**  Yes

**Roles Required**  None

**Response Type**  Array

1 Only those *Delivery Services* assigned to *Tenants* that are the requesting user’s *Tenant* or children thereof may be modified with this endpoint.
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 Delivery Service being modified</td>
</tr>
<tr>
<td>displayName</td>
<td>A string that is the Display Name</td>
</tr>
<tr>
<td>infoUrl</td>
<td>An optional string containing the Info URL</td>
</tr>
<tr>
<td>longDesc</td>
<td>An optional string containing the Long Description of this Delivery Service</td>
</tr>
<tr>
<td>longDesc1</td>
<td>An optional string containing the Long Description 2 of this Delivery Service</td>
</tr>
</tbody>
</table>

# 271: Request Example

```
PUT /api/1.5/deliveryservices/1/safe HTTP/1.1
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: mojolicious=...
Content-Length: 132

{
    "displayName": "test",
    "infoUrl": "this is not even a real URL",
    "longDesc": "longDesc1 is implicitly set to null in this example"
}
```

Response Structure

- **active** A boolean that defines *Active*.
- **anonymousBlockingEnabled** A boolean that defines *Anonymous Blocking*.
- **cacheurl** A *Cache URL Expression*
  
  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 *DNS TTL* - 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

\(^2\) If these fields are not present in the request body they are *implicitly set to null*. 
checkPath  A Check Path

consistentHashRegex  A Consistent Hashing Regular Expression
    New in version 1.4.

consistentHashQueryParams  An array of Consistent Hashing Query Parameters
    New in version 1.4.

deepCachingType  The Deep Caching setting for this Delivery Service
    New in version 1.3.

displayName  The Display Name

dnsBypassCname  A DNS Bypass CNAME

dnsBypassIp  A DNS Bypass IP

dnsBypassIp6  A DNS Bypass IPv6

dnsBypassTtl  The DNS Bypass TTL

dscp  A DSCP to be used within the Delivery Service

ecsEnabled  A boolean that defines the EDNS0 Client Subnet Enabled setting on this Delivery Service
    New in version 1.5.

edgeHeaderRewrite  A set of Edge Header Rewrite Rules

exampleURLs  An array of Example URLs

fqPacingRate  The Fair-Queuing Pacing Rate Bps
    New in version 1.3.

geoLimit  An integer that defines the Geo Limit

geoLimitCountries  A string containing a comma-separated list defining the Geo Limit Countries

geoLimitRedirectUrl  A Geo Limit Redirect URL

geoProvider  The Geolocation Provider

globalMaxMbps  The Global Max Mbps

globalMaxTps  The Global Max TPS

httpBypassFqdn  A HTTP Bypass FQDN

id  An integral, unique identifier for this Delivery Service

infoUrl  An Info URL

initialDispersion  The Initial Dispersion

ipv6RoutingEnabled  A boolean that defines the IPv6 Routing Enabled setting on this Delivery Service
lastUpdated The date and time at which this Delivery Service was last updated, in RFC 3339 format

logsEnabled A boolean that defines the Logs Enabled setting on this Delivery Service

longDesc The Long Description of this Delivery Service

longDesc1 The Long Description 2 of this Delivery Service

longDesc2 The Long Description 3 of this Delivery Service

matchList The Delivery Service’s Match List

  pattern A regular expression - the use of this pattern is dependent on the type field (backslashes are escaped)

  setNumber An integer that provides explicit ordering of Match List items - this is used as a priority ranking by Traffic Router, and is not guaranteed to correspond to the ordering of items in the array.

  type The type of match performed using pattern.

maxDnsAnswers The Max DNS Answers allowed for this Delivery Service

maxOriginConnections The Max Origin Connections

New in version 1.4.

midHeaderRewrite A set of Mid Header Rewrite Rules

missLat The Geo Miss Default Latitude used by this Delivery Service

missLong The Geo Miss Default Longitude used by this Delivery Service

multiSiteOrigin A boolean that defines the use of Use Multi-Site Origin Feature by this Delivery Service

orgServerFqdn The Origin Server Base URL

originShield A Origin Shield string

profileDescription The Description of the Profile with which this Delivery Service is associated

profileId The ID of the Profile with which this Delivery Service is associated

profileName The Name of the Profile with which this Delivery Service is associated

protocol An integral, unique identifier that corresponds to the Protocol used by this Delivery Service

qstringIgnore An integral, unique identifier that corresponds to the Query String Handling setting on this Delivery Service

rangeRequestHandling An integral, unique identifier that corresponds to the Range Request Handling setting on this Delivery Service
regexRemap A Regex Remap Expression

regionalGeoBlocking A boolean defining the Regional Geoblocking setting on this Delivery Service

remapText Raw Remap Text

signed true if and only if signingAlgorithm is not null, false otherwise

signingAlgorithm Either a Signing Algorithm or null to indicate URL/URI signing is not implemented on this Delivery Service

New in version 1.3.

sslKeyVersion This integer indicates the SSL Key Version

tenantId The integral, unique identifier of the Tenant who owns this Delivery Service

New in version 1.3.

trRequestHeaders If defined, this defines the Traffic Router Log Request Headers used by Traffic Router for this Delivery Service

New in version 1.3.

trResponseHeaders If defined, this defines the Traffic Router Additional Response Headers used by Traffic Router for this Delivery Service

New in version 1.3.

type The Type of this Delivery Service

typeId The integral, unique identifier of the Type of this Delivery Service

xmlId This Delivery Service’s xml_id

# 272: Response Example

```
HTTP/1.1 200 OK
Content-Encoding: gzip
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 10 Feb 2020 16:33:03 GMT; Max-Age=3600; HttpOnly
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Feb 2020 15:33:03 GMT
Content-Length: 853

{
    "alerts": [
        {
            "text": "Delivery Service safe update successful.",
            "level": "success"
        }
    ],
    "response": [
        ...
    ]
}
```

(continues on next page)
"active": true,
"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": 1,
"infoUrl": "this is not even a real URL",
"initialDispersion": 1,
"ipv6RoutingEnabled": true,
"lastUpdated": "2020-02-10 15:33:03+00",
"logsEnabled": true,
"longDesc": "longDesc1 is implicitly set to null in this example",
"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": 2,
"qstringIgnore": 0,
"rangeRequestHandling": 0,
"regexRemap": null,
"regionalGeoBlocking": false,
"remapText": null,
"routingName": "video",
"signed": false,
"sslKeyVersion": 1,
"tenantId": 1,
"type": "HTTP",
"typeId": 1,
"xmlId": "demol",
"exampleURLs": [
    "http://video.demol.mycdn.ciab.test",
    "https://video.demol.mycdn.ciab.test"
],
"deepCachingType": "NEVER",
"fqPacingRate": null,
"signingAlgorithm": null,
"tenant": "root",
"trResponseHeaders": null,
"trRequestHeaders": null,
"consistentHashRegex": null,
"consistentHashQueryParams": [
    "abc",
    "pdq",
    "xxx",
    "zyx"
],
"maxOriginConnections": 0,
"ecsEnabled": false
]}

deliveryservices/{{id}}/server_types/{{type}}/metric_types/
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.

5.1. Traffic Ops API 527
Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

Table 156: 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_type</td>
<td>yes</td>
<td>EDGE or MID.</td>
</tr>
<tr>
<td>metric_type</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 157: 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>
# 273: 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"
    }
  ]
}
```

deliveryservices/{{ID}}/servers

**Caution:** It’s often much easier to use `deliveryservices/{{xml_id}}/servers` instead.

**GET**

Retrieves properties of Edge-Tier servers assigned to a Delivery Service.

- **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 of the Delivery service for which servers will be displayed</td>
</tr>
</tbody>
</table>

Response Structure

cachegroup A string that is the *name of the Cache Group* to which the server belongs
cachegroupId An integer that is the *ID of 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 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
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 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 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 interface *interfaceName*

ip6Gateway The IPv6 gateway address of the server - applicable for the interface *interfaceName*

---

1 See the Wikipedia article on Out-of-Band Management for more information.
The IPv4 address of the server- applicable for the interface

The IPv4 gateway of the server- applicable for the interface

The IPv4 subnet mask of the server- applicable for the interface

The time and date at which this server was last updated, in an ISO-like format

The IPv4 address of the server’s management port

The IPv4 gateway of the server’s management port

The IPv4 subnet mask of the server’s management port

A user-entered reason why the server is in ADMIN_DOWN or OFFLINE status (will be empty if not offline)

The name of the Physical Location at which the server resides

An integral, unique identifier for the Physical Location at which the server resides

The Name of the Profile assigned to this server

A Description of the Profile assigned to this server

The ID of the Profile assigned to this server

A string indicating “rack” location

The human-readable name of the router

The human-readable name of the router port

The Status of the server

See also:

Health Protocol

An integral, unique identifier for the status of the server

See also:

Health Protocol

The default port on which the main application listens for incoming TCP connections - 80 in most cases

The name of the type of this server

An integral, unique identifier for the type of this server

true if the server has updates pending, false otherwise
# 274: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: MaIvaO8OSjysr4bCkuXFEMf3c6mOqgal1aM41HN/tcF2aaliXEmA5IrHB7DaqNX/2vGHLxvN+01FEAR/1RNqr1w==
X-Server-Name: traffic_ops_golang/
Date: Wed, 14 Nov 2018 21:28:23 GMT
Content-Length: 891

{
    "response": [
        {
            "cachegroupId": "CDN_in_a_Box_Edge",
            "cdnName": "CDN-in-a-Box",
            "domainName": "infra.ciab.test",
            "guid": null,
            "hostName": "edge",
            "httpsPort": 443,
            "id": 10,
            "iloIpAddress": "",
            "iloIpGateway": "",
            "iloIpNetmask": "",
            "iloPassword": "",
            "iloUsername": "",
            "interfaceMtu": 1500,
            "interfaceName": "eth0",
            "ip6Address": "fc01:9400:1000:8::100",
            "ip6Gateway": "fc01:9400:1000:8::1",
            "ipAddress": "172.16.239.100",
            "ipGateway": "172.16.239.1",
            "ipNetmask": "255.255.255.0",
            "lastUpdated": "2018-11-14 21:08:44+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,
        }
    ]
}
```
Caution: This endpoint may not work as advertised, and its use is therefore discouraged!

GET

Retrieves properties of Edge-Tier cache servers eligible for assignment to a particular Delivery Service. Eligibility is determined based on the following properties:

- The name of the server’s Type must match one of the glob patterns EDGE*, ORG*
- The server and Delivery Service must belong to the same CDN
- If the Delivery Service has Required Capabilities, an Edge-tier cache server must have all of those defined capabilities

Auth. Required  Yes
Roles Required  None
Response Type Array

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 of the Delivery service for which servers will be displayed</td>
</tr>
</tbody>
</table>
Response Structure

- **cachegroup**: A string which is the *Name of the Cache Group* to which the server belongs.
- **cachegroupId**: An integer that is the *ID of 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 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.
- **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 MTU to configure for *interfaceName*.
- **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*.

---

1 See the Wikipedia article on Out-of-Band Management for more information.
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 OF_FLINE 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 The ID of 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

# 275: Response Example

```json
{
  "response": [
    {
      "cachegroup": "CDN_in_a_Box_Edge",
      "cachegroupId": 7,
      "cdnId": 2,
      "cdnName": "CDN-in-a-Box",
      "domainName": "infra.ciab.test",
      ...
    }
  ]
}
```

(continues on next page)
deliveryservices/{{ID}}/state

Deprecated since version ATCv4.

GET

Retrieves the fail-over state for a Delivery Service.

Auth. Required  Yes
Roles Required  None

Response Type  Object

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 Delivery Service being inspected</td>
</tr>
</tbody>
</table>

Response Structure

enabled  true if failover has been enabled for this Delivery Service, false otherwise
ailover  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

type  The ‘type’ of the Cache Group identified by location

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

# 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

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
deliverservices/{{ID}}/unassigned_servers

**Danger:** 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 cache servers not* assigned to a *Delivery Service*.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”¹

**Response Type** Array

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.
Request Structure

Table 161: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>An integral, unique identifier for a Delivery Service</td>
</tr>
</tbody>
</table>

Response Structure

cachegroup  A string which is the Name of the Cache Group to which the server belongs

cachegroupId  An integer that is the ID of the Cache Group to which the server belongs

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

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 Name of the Profile assigned to this server

**profileDesc**  A Description of the Profile assigned to this server

**profileId**  The ID of 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 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

---

# 277: Response Example

```json
{
    "response": [
        {
            "cachegroup": "us-il-chicago",
            "cachegroupId": "3",
            "cdnId": "3",
            "cdnName": "CDN-1",
            "domainName": "chi.kabletown.net",
            "guid": "null",
            "hostName": "atsec-chi-00",
            "id": "19",
            "iloIpAddress": "172.16.2.6",
            "iloIpGateway": "172.16.2.1"
        }
    ]
}
```

(continues on next page)
deliveryservices_regexes

GET

Retrieves routing regular expressions for all Delivery Services.

Auth. Required Yes

Roles Required None

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

# 278: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: +2MI+Q/NJqTiz1MR/MhPAL+yu6/z/Yqvo5fD08F583RMOmK6dX/A14wARbEG+0QaJNqSCRPsiLVATuSrnppCMA==
```

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 rewrite rule to be used at the Edge tier

**headerRewriteMid** An optional string containing a header rewrite rule to be used at the Mid tier

**headerRewriteRedirectRouter** 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

e-mailTo The email to which the Delivery Service request will be sent
# 279: Request Example

```
{
  "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": "less than 50",
    "rateLimitingTPS": "no more than 5000",
    "overflowService": null,
    "headerRewriteEdge": "",
    "headerRewriteMid": "",
    "headerRewriteRedirectRouter": "",
    "notes": ""
  }
}
```

Response Structure

# 280: Response Example

```
{
  "alerts": [{
    "level": "success",
    "text": "Delivery Service request sent to foo@bar.com."
  }]
}
```

deliveryservices_required_capabilities

New in version 1.4.
GET

Gets all associations of *Server Capability* to *Delivery Services*.

**Auth. Required**  Yes

**Roles Required**  None

**Response Type**  Array

### Request Structure

Table 162: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>delivery-ServiceID</td>
<td>no</td>
<td>Filter <em>Server Capability</em> associations by <em>Delivery Service</em> integral, unique identifier</td>
</tr>
<tr>
<td>xmlID</td>
<td>no</td>
<td>Filter <em>Server Capability</em> associations by <em>Delivery Service</em> xml_id</td>
</tr>
<tr>
<td>required-Capability</td>
<td>no</td>
<td>Filter <em>Server Capability</em> associations by <em>Server Capability</em> name</td>
</tr>
<tr>
<td>orderBy</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit.</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n^{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>
# 281: Request Example

```plaintext
GET /api/1.4/deliveryservices_required_capabilities HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

## Response Structure

- **deliveryServiceID**  The associated Delivery Service’s integral, unique identifier
- **xmlID**  The associated Delivery Service’s xml_id
- **lastUpdated**  The date and time at which this association between the Delivery Service and the Server Capability was last updated, in an ISO-like format
- **requiredCapability**  The Server Capability’s name

# 282: Response Example

```json
{
    "response": [
        {
            "deliveryServiceID": 1,
            "lastUpdated": "2019-10-07 22:05:31+00",
            "requiredCapability": "ram",
            "xmlId": "example_ds-1"
        },
        {
            "deliveryServiceID": 2,
            "lastUpdated": "2019-10-07 22:05:31+00",
            "requiredCapability": "disk",
            (continues on next page)
        }
    ]
}
```

(continues on next page)
POST

Associates a *Server Capability* with a *Delivery Service*.

**Auth. Required**  Yes

**Roles Required**  “admin” or “operations”

**Response Type**  Object

**Note:**  A *Server Capability* can only be made required on a *Delivery Service* if its associated Servers already have that *Server Capability* assigned.

### Request Structure

- **deliveryServiceID**  The integral, unique identifier of the *Delivery Service* to be associated
- **requiredCapability**  The name of the *Server Capability* to be associated

#### # 283: Request Example

```
POST /api/1.4/deliveryservices_required_capabilities HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 56
Content-Type: application/json

{
    "deliveryServiceID": 1,
    "requiredCapability": "disk"
}
```

### Response Structure

- **deliveryServiceID**  The newly associated *Delivery Service*’s integral, unique identifier
**lastUpdated** The date and time at which this association between the *Delivery Service* and the *Server Capability* was last updated, in an ISO-like format

**requiredCapability** The newly associated *Server Capability’s name*

### # 284: 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, DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019
  17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: eQrl48zWids0kDpfCYmmtYMpegjnFxfOVvlBYxxLSfp7P7p6cWX4uiC+/Cfh2X9i3G+MQ36eH95gukJqOBOGbQ==
X-Server-Name: traffic_ops_golang/
Date: Mon, 07 Oct 2019 22:15:11 GMT
Content-Length: 287

{
    "alerts": [
        {
            "level": "success",
            "text": "deliveryservice.RequiredCapability was created."
        }
    ],
    "response": {
        "deliveryServiceID": 1,
        "lastUpdated": "2019-10-07 22:15:11+00",
        "requiredCapability": "disk"
    }
}
```

### DELETE

Dissociate a *Server Capability* from a *Delivery Service*.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** undefined
Request Structure

**deliveryServiceID** The integral, unique identifier of the *Delivery Service* from which a *Server Capability* will be dissociated

**requiredCapability** The name of the *Server Capability* to dissociate

# 285: Request Example

```
POST /api/1.4/deliveryservices_required_capabilities HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 56
Content-Type: application/json

{
    "deliveryServiceID": 1,
    "requiredCapability": "disk"
}
```

Response Structure

# 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,DELETE
Access-Control-Allow-Origin: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: 9eQr148zWids0kDpfCYmmtYMpegjnFxfOVv1BYxxLSfp7P7p6oWX4uiC/+Cfh2X9i3G+MQ36ehH95gukJqOB0GbQ==
X-Server-Name: traffic_ops_golang/
Date: Mon, 07 Oct 2019 22:15:11 GMT
Content-Length: 127

{
    "alerts": [
        {
            "level": "success",
            "text": "deliveryservice.RequiredCapability was deleted."
        }
    ]
}
```
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
cert 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
xml_id 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

# 287: Request Example

```json
POST /api/1.4/deliveryservices/sslkeys/add HTTP/1.1
Host: trafficops.infra.ciab.test
Content-Type: application/json

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

Response Structure

# 288: Response Example

```
HTTP/1.1 200 OK
Content-Type: application/json

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

deliveryservices/sslkeys/generate

**POST**

Generates 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** 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

---

#### # 289: 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

#### # 290: Response Example

```
HTTP/1.1 200 OK
Content-Type: application/json

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

---

**deliveryservices/sslkeys/generate/letsencrypt**

New in version 1.5.

**POST**

Generates an SSL certificate and private key using Let’s Encrypt for a Delivery Service

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Object (string)
Request Structure

**key**  The xml_id of the *Delivery Service* for which keys will be generated

**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

**hostname**  The desired hostname of the *Delivery Service*

---

**Note:**  In most cases, this must be the same as the *Example URLs*.

**cdn**  The name of the CDN of the *Delivery Service* for which the certs will be generated

---

# 291: Request Example

```
POST /api/1.5/deliveryservices/sslkeys/generate/letsencrypt HTTP/1.1
Content-Type: application/json

{
    "key": "ds-01",
    "version": "3",
    "hostname": "tr.ds-01.ott.kabletown.com",
    "cdn": "test-cdn"
}
```

---

Response Structure

---

# 292: Response Example

```
{
    "alerts": [{
        "level": "success",
        "text": "Beginning async call to Let's Encrypt for ds-01. This may take a few minutes."
    }]
}
```

deliveryservices/{{xml_id}}/servers

**POST**

Assigns *cache servers* to a *Delivery Service*.

**Auth. Required**  Yes

**Roles Required**  “admin” or “operations”\footnote{Users can only assign servers to *Delivery Services* that are visible to their *Tenant*.}
Response Type  Object

Request Structure

Table 163: 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

# 293: 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

# 294: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: zTpLrWlLM4xRsm8m1BQFB5KzT478AjloSyXHgtyWhebcV1YIwW1tmkjr0HFgc3GMGZ0Dt+fyzkOYy5yBtJw==
X-Server-Name: traffic_ops_golang/
Date: Tue, 20 Nov 2018 15:21:50 GMT
```

(continues on next page)
deliveryservices/{{xml_id}}/urisignkeys

DELETE

Deletes URISigning objects for a Delivery Service.

Auth. Required Yes
Roles Required admin
Response Type undefined

Request Structure

Table 164: Request Path 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 Structure

TBD

GET

Retrieves one or more URISigning objects for a delivery service.

Auth. Required Yes
Roles Required admin
Response Type undefined

1 URI Signing Keys can only be created, viewed, deleted, or modified on Delivery Services that either match the requesting user’s Tenant or are descendants thereof.

5.1. Traffic Ops API
Request Structure

Table 165: 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 Structure

Issuer a string describing the issuer of the URI signing object. Multiple URISigning objects may be returned in a response, see example

renewal_kid a string naming the jwt key used for renewals

keys json array of jwt symmetric keys

alg this parameter repeats for each jwt key in the array and specifies the jwa encryption algorithm to use with this key, RFC 7518

kid this parameter repeats for each jwt key in the array and specifies the unique id for the key as defined in RFC 7516

kty this parameter repeats for each jwt key in the array and specifies the key type as defined in RFC 7516

k this parameter repeats for each jwt key in the array and specifies the base64 encoded symmetric key see RFC 7516

# 295: Response Example

```
{
    "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": "fZBpDBNbk2GqhqwoB_-
DGBAsBxqZVix04rIoLJ7p_R1E"
            }
        ]
    }
}
```
**POST**

Assigns URISigning objects to a delivery service.

**Auth. Required** Yes

**Roles Required** admin

**Response Type** undefined

### Request Structure

<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>

**Issuer** a string describing the issuer of the URI signing object. Multiple URISigning objects may be returned in a response, see example

**renewal_kid** a string naming the jwt key used for renewals

**keys** json array of jwt symmetric keys

**alg** this parameter repeats for each jwt key in the array and specifies the jwa encryption algorithm to use with this key, [RFC 7518](https://tools.ietf.org/html/rfc7518)

**kid** this parameter repeats for each jwt key in the array and specifies the unique id for the key as defined in [RFC 7516](https://tools.ietf.org/html/rfc7516)

**kty** this parameter repeats for each jwt key in the array and specifies the key type as defined in [RFC 7516](https://tools.ietf.org/html/rfc7516)

**k** this parameter repeats for each jwt key in the array and specifies the base64 encoded symmetric key see [RFC 7516](https://tools.ietf.org/html/rfc7516)

### #296: Request Example

```json
{
   "Kabletown URI Authority": {
      "renewal_kid": "Second Key",
      "keys": [
         {
            "alg": "HS256",
            "kid": "First Key",
            "kty": "oct",
            "k": "Kh_RkUMj-fzbD37qBnDf_3e_RvQ3RP9PaSmVEpE24AM"
         }
      ]
   }
}
```

(continues on next page)
PUT

updates URISigning objects on a delivery service.

Auth. Required  Yes
Roles Required  admin
Response Type  undefined

Request Structure

<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>

Issuer  a string describing the issuer of the URI signing object. Multiple URISigning objects may be returned in a response, see example

renewal_kid  a string naming the jwt key used for renewals

keys  json array of jwt symmetric keys

alg  this parameter repeats for each jwt key in the array and specifies the jwa encryption algorithm to use with this key, RFC 7518

kid  this parameter repeats for each jwt key in the array and specifies the unique id for the key as defined in RFC 7516

kty  this parameter repeats for each jwt key in the array and specifies the key type as defined in RFC 7516

k  this parameter repeats for each jwt key in the array and specifies the base64 encoded symmetric key see RFC 7516
# 297: Request Example

```json
{
    "Kabletown URI Authority": {
        "renewal_kid": "Second Key",
        "keys": [
            {
                "alg": "HS256",
                "kid": "First Key",
                "kty": "oct",
                "k": "Kh_RkUMj-fzbD37qBnDf_3e_RvQ3R9PaSmVEpE24AM"
            },
            {
                "alg": "HS256",
                "kid": "Second Key",
                "kty": "oct",
                "k": "fZBpDBNbk2GqhwoB_DGBAsBxqQZix04rIoLJ7p_R1E"
            }
        ]
    }
}
```

deliveryservices/xmlId/{{XMLID}}/sslkeys

GET

Retrieves SSL keys for a Delivery Service.

- **Auth. Required**: Yes
- **Roles Required**: None
- **Response Type**: Object

**Request Structure**

Table 168: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLID</td>
<td>The <code>xml_id</code> of the desired Delivery Service</td>
</tr>
</tbody>
</table>
Table 169: 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`.

- **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  

---

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.
country An optional field which, if present, contains the country entered by the user when generating the SSL certificate¹

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 certificate¹

state An optional field which, if present, contains the state entered by the user when generating certificate¹

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

expiration The expiration date of the certificate for the Delivery Service in RFC 3339 format

New in version 1.5.

# 298: Response Example

```
HTTP/1.1 200 OK
Content-Type: application/json

{  "response": {
   "certificate": {
      "crt": "crt",
      "key": "key",
      "csr": "csr"
   },
   "deliveryservice": "my-ds",
   "cdn": "qa",
   "businessUnit": "CDN_Eng",
   "city": "Denver",
   "organization": "KableTown",
   "hostname": "foober.com",
   "country": "US",
   "state": "Colorado",
   "version": "1",
   "expiration": "2020-08-18T13:53:06Z"
  }
}
```

deliveryservices/xmlId/{{xmlid}}/sslkeys/delete

5.1. Traffic Ops API
GET

Auth. Required  Yes
Roles Required  “admin” or “operations”
Response Type  Object (string)

Request Structure

Table 170: 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 171: 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

# 299: 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>
<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])

# 300: Response Example

```json
{
  "response": { 
    "key9": "ZvVQNYpPVQWQV8tjQnU16osm4y7xK4zd",
    "key6": "JhQdpw5X9c8TqHfgezCm0bqb9QPASWL",
    "key8": "ySXdp1T8IeDEE10CMftz2b9Eiw_20wq",
    "key0": "D4AYzJAE2nYisA9MxMtY03TPDChji9C",
    "key3": "W90YH1Gc_kY1yw5_10Lrkv9J0zSiNeI",
    "key12": "ZbtMb3mrKqfs8hnx9_xWBIP_0PW1Upzc",
    "key2": "0ggE0D07sUsugIQemZbwMt0tNCwBlsf",
    "key4": "aFJ2Gb7a7txmVB8uv7T9S60aM13ycpGf",
    "key1": "wnWNRlMczl04C7EFPtqHd0xUMQyNPhA",
    "key11": "k6HMz1BH1x6htKkyqRFfWQhAndQqe50e",
    "key10": "zYONfdD7fGYKj4kLwIjxU918c0U00d",
    "key15": "3360cGaiUp_layZMc_qH2eJbazzTQh",
    "key5": "S1wv3G0hWN7E9wSwPFj18qE4M07sfXN",
    "key13": "SqQKBR6LqEOzp8Aew2UCVtBcw_8YFc1g",
    "key14": "DtXae8snw04YhT0kNoK8hu2G3P9WRpQJ",
    "key7": "cmK0IxXGAXxUMdCsWvnGLoIMGmNiuT5I"
  }
}
```

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 173: 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 to which keys will be copied</td>
</tr>
<tr>
<td>copyFrom_xml_id</td>
<td>The xml_id of the Delivery Service from which keys will be copied</td>
</tr>
</tbody>
</table>

Response Structure

# 301: Response Example

```json
{
  "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 174: 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 Delivery Service</td>
</tr>
</tbody>
</table>

Response Structure

# 302: Response Example

```json
{
  "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 175: Request Query Parameters

<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</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</td>
</tr>
</tbody>
</table>

# 303: 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

# 304: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: J7sK8PohQWyTpTrMjjrWd1JwPj+Zyep/xutM25uVosL6cHg13onXa6VMMyOC5Y3vd9r5KLES8rTgR+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,
}
```
POST

Assign a set of one or more servers to a Delivery Service

Auth. Required  Yes
Roles Required  “admin” or “operations”\(^2\)
Response Type  Object

Request Structure

- **dsId**  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`

# 305: 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

- **dsId**  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

\(^2\) 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.
servers  An array of integral, unique identifiers for servers which have been assigned to the Delivery Service identified by deliveryService

# 306: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512:D+HhGhoxxzavka9vZIStoaOZUpX23nz7zZnMbPFHNRO3MawyEaSb3GVUHQyCv6sDgwhpZZjRggDmct...
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 176: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Filter for <em>Divisions</em> having this integral, unique identifier</td>
</tr>
<tr>
<td>name</td>
<td>Filter for <em>Divisions</em> with this name</td>
</tr>
<tr>
<td>orderby</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>
<tr>
<td>sortOrder</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</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

---

# 307: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: SLKi9RHa67sGoSz62IDcQsk7KZjTXKfonqMoCUPPXGcNuhBsvvUjclG7KkwK8X1Ny16geMx2Bd8Hm...3dQ75GA==
X-Server-Name: traffic_ops_golang/
Date: Thu, 29 Nov 2018 19:44:03 GMT
Content-Length: 139

{
    "response": [
        {
            "id": 1,
            ...
        }
    ]
}
```
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

# 308: 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
# 309: 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: +pJm4c3O+JTaSXNt+LP+u240Ba/SsvSSDOQ4rDc6hcy20FIL+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.

Auth. Required  Yes

Roles Required  None

Response Type  Array
### Request Structure

#### Table 177: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>orderBy</td>
<td>no</td>
<td>Choose the ordering of the results - must be the name of one of the fields of the response array</td>
</tr>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

#### Table 178: 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>

#### # 310: Request Example

```
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

- **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

#### # 311: 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: *
```
PUT

Updates a specific Division

Auth. Required Yes

Roles Required “admin” or “operations”

Request Structure

Table 179: 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>
<tr>
<td>name</td>
<td>The new name of the Division</td>
</tr>
</tbody>
</table>

# 312: Request Example

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

{"name": "quest"}
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

# 313: Response Example

```json
{
    "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 ATCv4.

**POST**

Creates a new *Region* within the specified *Division*.

  **Auth. Required**  Yes

  **Roles Required**  “admin” or “operations”
Response Type  Object

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 division in which to create the new region</td>
</tr>
</tbody>
</table>

name  The name of the new region

# 314: Request Example

```
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

# 315: 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, 06 Dec 2018 00:03:36 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
```

(continues on next page)
federation_resolvers

GET

Retrieves *Federation* Resolvers.

Auth. Required  Yes
Roles Required  None
Response Type  Array
Request Structure

Table 181: 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 Federation Resolver identified by this integral, unique identifier</td>
</tr>
<tr>
<td>ipAddress</td>
<td>no</td>
<td>Return only the Federation Resolver(s) that has/have this IP Address</td>
</tr>
<tr>
<td>type</td>
<td>no</td>
<td>Return only the Federation Resolvers of this <em>Type</em></td>
</tr>
<tr>
<td>orderBy</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

# 316: Request Example

GET /api/1.4/federation_resolvers?type=RESOLVE6 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.63.0
Accept: */*
Cookie: mojolicious=...

Response Structure

**id**  The integral, unique identifier of the resolver

**ipAddress**  The IP address or CIDR-notation subnet of the resolver - may be IPv4 or IPv6

**lastUpdated**  The date and time at which this resolver was last updated, in an ISO-like format

New in version 1.4.

**type**  The *Type* of the resolver

# 317: 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)
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

**Caution:** This field should only ever be an identifier for one of the `Types”` “RESOLVE4” or “RESOLVE6”, but there is no protection for this built into Traffic Ops and therefore any valid `Type` identifier will be silently accepted by Traffic Ops and so care should be taken to ensure that these `Types` are properly identified. If any `Type` besides “RESOLVE4” or “RESOLVE6” is identified, the resulting resolver will not work.

See also:

`types` is the endpoint that can be used to determine the identifier for various `Types`
# 318: Request Example

```
POST /api/1.4/federation_resolvers HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.63.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 36
Content-Type: application/json

{
    "ipAddress": "::1/1",
    "typeId": 37
}
```

## Response Structure

- **id**: The integral, unique identifier of the resolver
- **ipAddress**: The IP address or CIDR-notation subnet of the resolver - may be IPv4 or IPv6
- **type**: The *Type* of the resolver
- **typeId**: The integral, unique identifier of the *Type* of the resolver

---

# 319: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: e9D8JNrQb64xpuDwcBwbISSWUKDGCL2137NuDXsXPYof2EqmeHondD8NzxDSwWNj8d9B9DXpZ DbRU
X-Server-Name: traffic_ops_golang/
Date: Wed, 06 Nov 2019 00:00:40 GMT
Content-Length: 153

```
{
    "alerts": [
        {
            "text": "Federation Resolver created [ IP = ::1/1 ] with id: 1",
            "level": "success"
        }
    ]
}
```
DELETE

Deletes a federation resolver.

New in version 1.5.

Auth. Required Yes
Roles Required “admin”
Response Type Object

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>Integral, unique identifier for the federation resolver to be deleted</td>
</tr>
</tbody>
</table>

# 320: Request Example

```
DELETE /api/1.5/federation_resolvers?id=4  HTTP/1.1
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: mojolicious=...
Content-Length: 0
```

Response Structure

- **id**: The integral, unique identifier of the resolver
- **ipAddress**: The IP address or CIDR-notation subnet of the resolver - may be IPv4 or IPv6
- **type**: The Type of the resolver
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-Encoding: gzip
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; HttpOnly
Whole-Content-Sha512: 2v4LYQdRVhaFJVd86Iv1BWVYzNPS1zpQ222bUB7Zz+Ss8A48FNyHZjP1q5a+a4g9KAQCTUIytWnI0k
X-Server-Name: traffic_ops_golang/
Date: Fri, 08 Nov 2019 23:19:01 GMT
Content-Length: 161

{   "alerts": [
    {
      "text": "Federation resolver deleted [ IP = 1.2.6.4/22 ] with id: 4",
      "level": "success"
    }
  ],
  "response": {
    "id": 4,
    "ipAddress": "1.2.6.4/22",
    "type": "RESOLVE6"
  }
}

federation_resolvers/{{ID}}

Deprecated since version 1.5.

DELETE

Deletes a federation resolver.

Auth. Required  Yes

Roles Required  “admin”

Response Type  Object

Changed in version 1.5: Older versions of this endpoint did not return a response object (i.e. it was undefined).
Request Structure

Table 183: 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 federation resolver to be deleted</td>
</tr>
</tbody>
</table>

# 322: Request Example

```
DELETE /api/1.4/federation_resolvers/2 HTTP/1.1
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: mojolicious=...
Content-Length: 0
```

Response Structure

- **id** The integral, unique identifier of the resolver
- **ipAddress** The IP address or CIDR-notation subnet of the resolver - may be IPv4 or IPv6
- **type** The Type of the resolver

# 323: 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-Encoding: gzip
Content-Type: application/json
Cookie: mojolicious=; Path=/; HttpOnly
Whole-Content-Sha512: Gpvrv90AIWw1HrLsPyqm0Jhn+bDMZ8X/MwLXH8iXyi9DFsLdq5dLsi8LjNM7fiXP0j51bRJxpNH4dKxsvSrTg==
X-Server-Name: traffic_ops_golang/
Date: Sat, 09 Nov 2019 07:30:09 GMT
Content-Length: 215

{ "alerts": [
   
   "text": "Federation resolver deleted [ IP = 1.2.6.4/22 ] with id: 2",
   "level": "success"
],
```
(continues on next page)
federations

GET

Retrieves a list of Federation mappings (i.e. 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 CNAME to one or more Resolvers

  cname The actual CNAME used by the Federation

  resolve4 An array of IPv4 addresses (or subnets in CIDR notation) capable of resolving the Federation's CNAME

  resolve6 An array of IPv6 addresses (or subnets in CIDR notation) capable of resolving the Federation's CNAME

  ttl The TTL of the CNAME in hours
# 324: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: d6Llm5qNc2sfgVH9ImW7hA4wvtBuq6EzUmpJf805kB0k6v2WysNgFEWK4hBXNdAYkr8hYuKPrwDy3
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 method of 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:

---

586 Chapter 5. APIs
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

Changed in version 1.4: Prior to API version 1.4, the request body had to be wrapped in a top-level `federations` key, as can be seen in the Legacy Request example. That behavior is still supported but no longer necessary.

#### # 325: Legacy Request

```json
{
  "federations": [
    {
      "deliveryService": "demo1",
      "mappings": {
        "resolve4": ["0.0.0.0"],
        "resolve6": ["::1"]
      }
    }
  ]
}
```

The request payload is an array of objects that describe Delivery Service Federation Resolver mappings. Each object in the array must be in the following format.

- **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 (or subnets in CIDR notation) to use as Federation Resolvers:
  - **resolve4** An array of IPv4 addresses (or subnets in CIDR notation) that can resolve the Delivery Service’s Federation.
  - **resolve6** An array of IPv6 addresses (or subnets in CIDR notation) that can resolve the Delivery Service’s Federation.

#### # 326: Request Example

```
POST /api/1.4/federations HTTP/1.1  
Host: trafficops.infra.ciab.test  
User-Agent: curl/7.47.0  
Accept: */*  
Cookie: mojolicious=...  
Content-Length: 118  
Content-Type: application/json
```

(continues on next page)
Response Structure

# 327: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: B7TSUOYZPRPyi3mVy+CuxiXR5k/d0s07w4i6kYzpWS+YL79juEfkusqfedaYG/kMA809XbjkWRjcBAdxOVrdTQ==
X-Server-Name: traffic_ops_golang/
Date: Wed, 23 Oct 2019 22:28:02 GMT
Content-Length: 152

{
    "alerts": [
        {
            "text": "admin successfully created federation resolvers."
        }
    ],
    "response": "admin successfully created federation resolvers."
}
```

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

# 328: Request Example

```
DELETE /api/1.4/federations HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

# 329: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: fd7P45mIiHuYqZZW6+8K+YjY1Pe504Aaw4J4Zp9AhrqLX72ERytTqWtAp1msutzNSRUdUSC72+odNP X-Server-Name: traffic_ops_golang/
Date: Wed, 23 Oct 2019 23:34:53 GMT
Content-Length: 184

{
  "alerts": [
    {
      "text": "admin successfully deleted all federation resolvers: [ 8.8.8.8 ]",
      "level": "success"
    }
  ],
  "response": "admin successfully deleted all federation resolvers: [ 8.8.8.8 ]"
}
```

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

Changed in version 1.4: Prior to API version 1.4, the request body had to be wrapped in a top-level `federations` key, as can be seen in the Legacy Request example. That behavior is still supported but no longer necessary.

# 330: Legacy Request

```
{
    "federations": [{
        "deliveryService": "demo1",
        "mappings": {
            "resolve4": ["0.0.0.0"],
            "resolve6": ["::1"]
        }
    }]
}
```

The request payload is an array of objects that describe Delivery Service Federation Resolver mappings. Each object in the array must be in the following format.

- **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 (or subnets in CIDR notation) to use as Federation Resolvers
  - **resolve4**  An array of IPv4 addresses (or subnets in CIDR notation) that can resolve the Delivery Service’s Federation
  - **resolve6**  An array of IPv6 addresses (or subnets in CIDR notation) that can resolve the Delivery Service’s Federation

# 331: Request Example

```
PUT /api/1.4/federations HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 95
Content-Type: application/json

[{ "mappings": {
    "resolve4": ["8.8.8.8"],
    "resolve6": []
  }
},
```

(continues on next page)
<Response Structure>

# 332: 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: dQ5AvQULhc254zQwgUpB11/CHbLr/c1ktks0Ju9f1B4xIffbo3puFNN9zaEa21iz0lBvHP/pGfUqisD3QHA==
X-Server-Name: traffic_ops_golang/
Date: Wed, 23 Oct 2019 23:22:03 GMT
Content-Length: 258
Content-Type: application/json

{
  "alerts": [
    {
      "text": "admin successfully deleted all federation resolvers: [ 8.8.8.8 ]",
      "level": "success"
    },
    {
      "text": "admin successfully created federation resolvers.",
      "level": "success"
    }
  ],
  "response": "admin successfully created federation resolvers."
}
```

<federations/{ID}/deliveryservices>

<GET>

Retrieves Delivery Services assigned to a Federation.

Auth. Required Yes
Roles Required None

5.1. Traffic Ops API
Response Type  Array

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 for the federation to be inspected</td>
</tr>
</tbody>
</table>

Table 185: 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>Show only the Delivery Service identified by this integral, unique identifier</td>
</tr>
<tr>
<td>orderby</td>
<td>no</td>
<td>Choose the ordering of the results - must be the name of one of the fields of the response array</td>
</tr>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

# 333: 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

<table>
<thead>
<tr>
<th>cdn</th>
<th>The CDN to which this Delivery Service Belongs</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The integral, unique identifier for the Delivery Service</td>
</tr>
<tr>
<td>type</td>
<td>The routing type used by this Delivery Service</td>
</tr>
<tr>
<td>xmlId</td>
<td>The ‘xml_id’ which uniquely identifies this Delivery Service</td>
</tr>
</tbody>
</table>

# 334: Response Example

```
HTTP/1.1 200 OK
access-control-allow-credentials: true
```
POST

Assigns one or more Delivery Services to a federation.

Auth. Required Yes
Roles Required “admin”
Response Type Object

Request Structure

Table 186: 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>
<tr>
<td>dsIds</td>
<td>An array of integral, unique identifiers for Delivery Services which will be assigned to this federation</td>
</tr>
<tr>
<td>replace</td>
<td>An optional boolean (default: false) which, if true, will cause any conflicting assignments already in place to be overridden by this request</td>
</tr>
</tbody>
</table>
Note: If replace is not given (and/or not true), then any conflicts with existing assignments will cause the entire operation to fail.

# 335: 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

# 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: 氵rVd0nx8G3bR18ub1zw6FTdmwQ7jer4zoqz0Zf5tC1ckrR0HEIOH1Azcmvv0FVE5I0oncHVnrYbza
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"
        }
    ]
}```
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 187: 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>

# 337: 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
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/j4gWX7ye0kGGakRRFEkK6ShxqXvCxE0dCTyu75qILPN2wSgr3FGQnp2Sq345sE7In9g==
content-length: 98

{
    "alerts": [
        {
            "level": "success",
            "text": "federation deliveryservice was deleted."
        }
    ]
}

federations/{{ID}}/federation_resolvers

GET

Retrieves federation resolvers assigned to a federation.

Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

Table 188: 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>
# 339: 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)

# 340: 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
type: application/json
date: Wed, 05 Dec 2018 00:49:50 GMT
server: Mojolicious (Perl)
set-cookie: mojolicious=...; expires=Wed, 05 Dec 2018 04:49:50 GMT;
path=/; HttpOnly
vary: Accept-Encoding
whole-content-sha512: csC18kE3YjiILHP1wmJg7V4h/
   XWY8HUMKypu2Wnde2g7HJ4gTY51HfjCSqhyKvIqJQ8R17uEqshF3Ey6xIMOX4A==
content-length: 63

{ "response": [
  {
    "ipAddress": "0.0.0.0",
    "type": "RESOLVE4",
    "id": 1
  }
]}
```
**POST**

Assigns one or more resolvers to a federation.

**Auth. Required** Yes  
**Roles Required** “admin”  
**Response Type** Object

### 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 for the federation for which resolvers will be retrieved</td>
</tr>
<tr>
<td>fedResolverIds</td>
<td>An array of integral, unique identifiers for federation resolvers</td>
</tr>
<tr>
<td>replace</td>
<td>An optional boolean (default: false) which, if true, will cause any conflicting assignments already in place to be overridden by this request</td>
</tr>
</tbody>
</table>

**Note:** If replace is not given (and/or not true), then any conflicts with existing assignments will cause the entire operation to fail.

### # 341: 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
# 342: 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
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+RxyFOtJkNH0kzYlYQBS+PjpxDeCgw1l9+0poW5dyHP95SwbNCA==
content-length: 148

{
  "alerts": [ 
    {
      "level": "success",
      "text": "1 resolver(s) were assigned to the test. quest. federation"
    }
  ],
  "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 190: 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>

Table 191: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userID</td>
<td>no</td>
<td>Show only the user that has this integral, unique identifier</td>
</tr>
<tr>
<td>role</td>
<td>no</td>
<td>Show only the users that have this role</td>
</tr>
<tr>
<td>orderBy</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n&lt;sup&gt;th&lt;/sup&gt; page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</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”

# 343: 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 00:31:34 GMT
```
server: Mojolicious (Perl)
set-cookie: mojolicious=...; expires=Wed, 05 Dec 2018 04:31:34 GMT;
˓→path=/; HttpOnly
vary: Accept-Encoding
whole-content-sha512: eQQoF2x1bK2I2oTja7zrt/
˓→F1kLzCgwpuU2zb2+rmIjHbHJ3MnmsSczSamIAAYtzs5gDaqcuUX1G352B8d7Bj82g==
content-length: 101

```json
{
    "response": [
        {
            "fullName": null,
            "email": null,
            "id": 2,
            "role": "admin",
            "company": null,
            "username": "admin"
        }
    ]
}
```

**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.

# 344: Request Example

```sh
POST /api/1.4/federations/1/users HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
```

(continues on next page)
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

# 345: 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
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: MvPmgOAs58aSGQv+9EilflgOexbaexg+qE2IPrQZX0H4iSX4JvEys9adbGE9a9yaLj9uUMxg77N6Z
content-length: 137

{
    "alerts": [
        {
            "level": "success",
            "text": "1 user(s) were assigned to the test.quest federation"
        }
    ],
    "response": {
        "userIds": [2],
        "replace": true
    }
}
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 Example**

```
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**

**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
```
hwinfo

 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 193: 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>An integral, unique identifier of a specific hwinfo object which will be retrieved</td>
</tr>
<tr>
<td>server-Host-Name</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>
<tr>
<td>orderBy</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return. Default if not specified is 1000.</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect.</td>
</tr>
</tbody>
</table>

Caution: The lastUpdated query parameter doesn’t seem to work properly, and its use is therefore discouraged.

# 348: Request Example

GET /api/1.3/hwinfo HTTP/1.1
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: mojolicious=...
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

Also, in addition to the regular `response` field and any and all `alerts`, this endpoint returns an extra top-level JSON key: `limit`.

- **limit**: The number of results to which the result was limited. Should be exactly as specified in the `Request Structure`.

# 349: Response Example

```json
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Encoding: gzip
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; Expires=Fri, 22 Nov 2019 20:28:07 GMT; Max-Age=3600; HttpOnly
X-Server-Name: traffic_ops_golang/
Date: Fri, 22 Nov 2019 19:28:07 GMT
Content-Length: 138

{
    "alerts": [
        {
            "text": "This endpoint is deprecated, and will be removed in the future",
            "level": "warning"
        }
    ],
    "response": [
        {
            "description": "quest",
            "lastUpdated": "2019-11-22 19:31:26+00",
            "serverHostName": "dns",
            "serverId": 1,
            "val": "test"
        }
    ],
    "limit": 1000
}
```
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, an LACP (Link Aggregation Control Protocol) 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”.

### # 350: 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",
    "dhcp": "no",
    "interfaceMtu": 1500,
    "ipAddress": "1.3.3.7",
    "ipNetmask": "255.255.255.255",
    "ipGateway": "8.0.0.8",
    "ip6Address": "1::3:3:7",
}
```

(continues on next page)

\(^1\) This optional key is required if and only if dhcp is “no”.

---

608 Chapter 5. APIs


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

# 351: Response Example

```json
{
  "alerts": [
    {
      "level": "success",
      "text": "Generate ISO was successful."
    }
  ],
  "response": {
    "isoURL": "https://some-weird-url.biz.co.uk/iso/test.quest-centos72.iso",
    "isoName": "test.quest-centos72.iso"
  }
}
```
GET

Retrieve content invalidation jobs.

**Auth. Required** Yes  
**Roles Required** None

**Response Type** Array

### Request Structure

#### Table 194: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>assetUrl</td>
<td>no</td>
<td>Return only invalidation jobs that operate on URLs by matching this regular expression</td>
</tr>
<tr>
<td>createdBy</td>
<td>no</td>
<td>Return only invalidation jobs that were created by the user with this username</td>
</tr>
<tr>
<td>deliveryService</td>
<td>no</td>
<td>Return only invalidation jobs that operate on the Delivery Service with this xml_id</td>
</tr>
<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>id</td>
<td>no</td>
<td>Return only the single invalidation job identified by this integral, unique identifier</td>
</tr>
<tr>
<td>keyword</td>
<td>no</td>
<td>Return only invalidation jobs that have this “keyword” - only “PURGE” should exist</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>

### # 352: Request Example

```
GET /api/1.4/jobs?id=3&dsId=1&userId=2 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: python-requests/2.20.1
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: mojolicious=...
```

---

1 When viewing content invalidation jobs, only those jobs that operate on a Delivery Service visible to the requesting user’s Tenant will be returned. Likewise, creating a new content invalidation job requires that the target Delivery Service is modifiable by the requesting user’s Tenant. However, when modifying or deleting an existing content invalidation job, the operation can be completed if and only if the requesting user’s Tenant is the same as the job’s Delivery Service’s Tenant or a descendant thereof, and if the requesting user’s Tenant is the same as the Tenant of the user who initially created the job or a descendant thereof.
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 of 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 a non-standard format

---

# 353: Response Example

```json
{
  "response": [
    {
      "assetUrl": "http://origin.infra.ciab.test/.*",
      "createdBy": "admin",
      "deliveryService": "demo1",
      "id": 3,
      "keyword": "PURGE",
      "parameters": "TTL:2h",
      "startTime": "2019-06-18 21:28:31+00"
    }
  ]
}
```

---

**POST**

New in version 1.4.
Creates a new content invalidation job.

Caution: Creating a content invalidation job immediately triggers a CDN-wide revalidation update. In the case that the global Parameter use_reval_pending has a value of exactly "0", this will instead trigger a CDN-wide “Queue Updates”. This means that content invalidation jobs become active immediately at their startTime - unlike most other configuration changes they do not wait for a Snapshot or a “Queue Updates”. Furthermore, if the global Parameter use_reval_pending is "0", this will cause all pending configuration changes to propagate to all cache servers in the CDN. Take care when using this endpoint.

Auth. Required Yes
Roles Required “operations” or “admin”
Response Type Object

Request Structure

deliveryService This should either be the integral, unique identifier of a Delivery Service, or a string containing an xml_id

startTime This can be a string in the legacy YYYY-MM-DD HH:MM:SS format, or a string in RFC 3339 format, or a string representing a date in the same non-standard format as the last_updated fields common in other API responses, or finally it can be a number indicating the number of milliseconds since the Unix Epoch (January 1, 1970 UTC). This date must be in the future, but unlike user/current/jobs (or the PUT method of this endpoint), it is not required to be within two days from the time of creation.

regex A regular expression that will be used to match the path part of URIs for content stored on cache servers that service traffic for the Delivery Service identified by deliveryService.

ttl Either the number of hours for which the content invalidation job should remain active, or a “duration” string, which is a sequence of numbers followed by units. The accepted units are:

- h gives a duration in hours
- m gives a duration in minutes
- s gives a duration in seconds
- ms gives a duration in milliseconds
- us (or µs) gives a duration in microseconds
- ns gives a duration in nanoseconds

These durations can be combined e.g. 2h45m specifies a TTL of two hours and forty-five minutes - however note that durations are always rounded up
to the nearest hour so that e.g. 121m becomes three hours. TTLs cannot ever be negative, obviously.

# 354: Request Example

```plaintext
POST /api/1.4/jobs HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: python-requests/2.20.1
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: mojolicious=...
Content-Length: 80
Content-Type: application/json

{
    "deliveryService": "demo1",
    "startTime": 1560893311219,
    "regex": "/.*/",
    "ttl": "121m"
}
```

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` of 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 a non-standard format

# 355: 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-Encoding: gzip
```

(continues on next page)
PUT

New in version 1.4.

Replaces an existing content invalidation job with a new one provided in the request. This method of editing a content invalidation job does not prevent the requesting user from changing fields that normally only have one value. Use with care.

**Caution:** Modifying a content invalidation job immediately triggers a CDN-wide revalidation update. In the case that the global Parameter `use_reval_pending` has a value of exactly "0", this will instead trigger a CDN-wide “Queue Updates”. This means that content invalidation jobs become active immediately at their `startTime` - unlike most other configuration changes they do not wait for a Snapshot or a “Queue Updates”. Furthermore, if the global Parameter `use_reval_pending` is "0", this will cause all pending configuration changes to propagate to all cache servers in the CDN. Take care when using this endpoint.

**Auth. Required** Yes
Roles Required  “operations” or “admin”\(^1\)

Response Type  Object

Request Structure

Table 195: 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 content invalidation job being modified</td>
</tr>
<tr>
<td>assetUrl</td>
<td></td>
<td>A regular expression - matching URLs will be operated upon according to keyword</td>
</tr>
<tr>
<td>createdBy</td>
<td></td>
<td>The username of the user who initiated the job(^2)</td>
</tr>
<tr>
<td>deliveryService</td>
<td></td>
<td>The xml_id of the Delivery Service on which this job operates(^2) - unlike POST request payloads, this cannot be an integral, unique identifier</td>
</tr>
<tr>
<td>id</td>
<td></td>
<td>An integral, unique identifier for this job(^2)</td>
</tr>
<tr>
<td>keyword</td>
<td></td>
<td>A keyword that represents the operation being performed by the job. It can have any (string) value, but the only value with any meaning to Traffic Control is: PURGE This job will prevent caching of URLs matching the assetUrl until it is removed (or its Time to Live expires)</td>
</tr>
<tr>
<td>parameters</td>
<td></td>
<td>A string containing space-separated key/value pairs - delimited by colons (:) representing parameters associated with the job. In practice, any string can be passed as a job’s parameters, but the only value with meaning is a single key/value pair indicated a TTL in hours in the format TTL:hours, and any other type of value may cause components of Traffic Control to work improperly or not at all.</td>
</tr>
<tr>
<td>startTime</td>
<td></td>
<td>This can be a string in the legacy YYYY-MM-DD HH:MM:SS format, or a string in RFC 3339 format, or a string representing a date in the same non-standard format as the last_updated fields common in other API responses, or finally it can be a number indicating the number of milliseconds since the Unix Epoch (January 1, 1970 UTC). This must be in the future, but only by no more than two days.</td>
</tr>
</tbody>
</table>

Note: Unlike in the payloads of POST requests to this endpoint, this must be a full URL regular expression, as it is not combined with the Origin Server Base URL of the Delivery Service identified by deliveryService.

\(^2\) This field must exist, but it must not be different than the same field of the existing job (i.e. as seen in a GET response)
# 356: Request Example

```plaintext
PUT /api/1.4/jobs?id=3 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: python-requests/2.20.1
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: mojolicious=...
Content-Length: 188
Content-Type: application/json

{
    "assetUrl": "http://origin.infra.ciab.test/.*",
    "createdBy": "admin",
    "deliveryService": "demo1",
    "id": 3,
    "keyword": "PURGE",
    "parameters": "TTL:360h",
    "startTime": "2019-06-20 18:33:40+00"
}
```

**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 of 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 a non-standard format

# 357: 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: *
```

(continues on next page)
DELETE

New in version 1.4.

Deletes a content invalidation job.

Tip: Content invalidation jobs that have passed their TTL are not automatically deleted - for record-keeping purposes - so use this to clean up old jobs that are no longer useful.

Caution: Deleting a content invalidation job immediately triggers a CDN-wide revalidation update. In the case that the global Parameter use_reval_pending has a value of exactly "0", this will instead trigger a CDN-wide “Queue Updates”. This means that content invalidation jobs become active immediately at their startTime - unlike most other configuration changes they do not wait for a Snapshot or a “Queue Updates”. Furthermore, if the global Parameter use_reval_pending is "0", this will cause all pending configuration changes to propagate to all cache servers in the CDN. Take care when using this endpoint.

Auth. Required Yes

Roles Required “operations” or “admin”
Response Type  Object

Request Structure

Table 196: 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 content invalidation job being modified</td>
</tr>
</tbody>
</table>

# 358: Request Example

```plaintext
DELETE /api/1.4/jobs?id=3  HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: python-requests/2.20.1
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: mojolicious=...
Content-Length: 0
```

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 of 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 a non-standard format

# 359: 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
```

(continues on next page)
Access-Control-Allow-Origin: *
Content-Encoding: gzip
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: FqfziXJYYWhb84Fac9+p4NEY3EsI1Yxe94wg/
 VOmlXk4R614SaPSh015CChPt/yT72MsWSETnIuRD9KtoK4I+w==
X-Server-Name: traffic_ops_golang/
Date: Tue, 18 Jun 2019 22:55:15 GMT
Content-Length: 234

{
  "alerts": [
    {
      "text": "Content invalidation job was deleted",
      "level": "success"
    }
  ],
  "response": {
    "assetUrl": "http://origin.infra.ciab.test/.*",
    "createdBy": "admin",
    "deliveryService": "demo1",
    "id": 3,
    "keyword": "PURGE",
    "parameters": "TTL:36h",
    "startTime": "2019-06-20 18:33:40+00"
  }
}

**jobs/**{{ID}}

Deprecated since version ATCv4: Use the GET method of **jobs** with the **id** query parameter instead.

**GET**

Get details about a specific content invalidation job.

- **Auth. Required** Yes
- **Roles Required** “operations” or “admin”
- **Response Type** Array

---

1 When viewing content invalidation jobs, only those jobs that operate on a **Delivery Service** visible to the requesting user’s **Tenant** will be returned.
Request Structure

Table 197: Request Path Parameters

<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>

# 360: Request Example

```plaintext
GET /api/1.4/jobs/3 HTTP/1.1
Host: trafficopsinfra.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 of 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 a non-standard format

# 361: 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: l7qvgOShdIFukHyOhhibes2BG6zJZ6RXTT70KABtI8b1y+cE4nxFq11T5OG5yXjKo69eTYOD7xUUDLq
X-Server-Name: traffic_ops_golang/
```
letsencrypt/autorenew

New in version 1.5.

**POST**

Generates an SSL certificate and private key using Let’s Encrypt for a Delivery Service

Auth. Required  Yes

Roles Required  “admin” or “operations”

Response Type  Object

**Request Structure**

No parameters available

**Response Structure**

- **LetsEncryptExpiration** A list of objects with information regarding certificate expiration for all Let’s Encrypt certificates
  - **XmlId** The Delivery Service’s uniquely identifying xml_id
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

Expiration  The expiration date of the certificate for the Delivery Service in RFC 3339 format

AuthType  The authority type of the certificate for the Delivery Service

Error  Any errors received in the renewal process

SelfSignedExpirations  A list of objects with information regarding certificate expiration for all self signed certificates

XmlId  The Delivery Service’s uniquely identifying xml_id

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

Expiration  The expiration date of the certificate for the Delivery Service in RFC 3339 format

AuthType  The authority type of the certificate for the Delivery Service

Error  Any errors received in the renewal process

OtherExpirations  A list of objects with information regarding certificate expiration for all other certificates

XmlId  The Delivery Service’s uniquely identifying xml_id

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

Expiration  The expiration date of the certificate for the Delivery Service in RFC 3339 format

AuthType  The authority type of the certificate for the Delivery Service

Error  Any errors received in the renewal process

# 362: Response Example

```
HTTP/1.1 200 OK
Content-Type: application/json

{ "response": {  "LetsEncryptExpirations": [   {     "XmlId": "demo2",     "Version": 1,   },   {     "XmlId": "demo3",     "Version": 2,   }  ] }
```

(continues on next page)
letsencrypt/dnsrecords

New in version 1.5.

**GET**

 Gets DNS challenge records for Let’s Encrypt DNS challenge for a specified FQDN.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Array

**Request Structure**

Table 198: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fqdn</td>
<td>no</td>
<td>Return only DNS challenge records for the specified FQDN</td>
</tr>
</tbody>
</table>

# 363: Request Example

```bash
GET /api/1.5/letsencrypt/dnsrecord?fqdn=_acme-challenge.demo1.example.com. HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...```

5.1. Traffic Ops API 623
Response Structure

fqdn  The FQDN for the TXT record location to complete the DNS challenge
record  The record provided by Let’s Encrypt to complete the DNS challenge

# 364: Response Example

```
HTTP/1.1 200 OK
Content-Type: application/json

{ "response": [
  {
   "fqdn": "_acme-challenge.demo1.example.com.",
   "record": "testRecord"
  }
]}
```

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

Fetched 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>
<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>
# 365: 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
Accept: */*
Cookie: 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

# 366: 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 15:11:38 GMT
Server: Mojolicious (Perl)
Set-Cookie: last_seen_log="2018-11-15%2015:11:38"; path=/; Max-Age=604800
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: 40dV+azaZ3b6F30y6YHvBv3H2a3ek2rdoxICupwaxQnj62pwYfb7YCM7Qhe30AItmB77Tbg91Ny27y
Content-Length: 357

{
    "response": [
        {
            "ticketNum": null,
            "level": "APICHANGE",
            "lastUpdated": "2018-11-14 21:40:06.493975+00",
            "user": "admin",
            "id": 444,
        }
    ]
}
```
logs/{{days}}/days

GET

Fetches a list of changes that have been made to the Traffic Control system in the past `days` days

**Auth. Required** Yes

**Roles Required** None

**Response Type** Array

### Request Structure

**Table 200: 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 201: 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>

# 367: 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

# 368: 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: Thu, 15 Nov 2018 15:19:10 GMT
Server: Mojolicious (Perl)
Set-Cookie: last_seen_log="2018-11-15%15:19:10"; path=/; Max-Age=604800
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: 40dV+azaZ3b6F30y6YHbV3H2a3ekZrdoxICupwaxQnj62pwYfb7YCM7Qhe3OItmB77Tbg9INy27y
Content-Length: 357

{ "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 | demol ]."
  },
  {
    "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"
  }
]
```
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

# 369: Response Example

```text
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=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: Ugdqe8GKSOExphwbDX/36Gli+2vBpuhtthpfYbJcAp7adox3MzmVRi2RxTDL5kwPewrcL1CQ88zGITskh0sc9g==
```
origins

New in version 1.3.

GET

Gets all requested origins.

Auth. Required  Yes
Roles Required  None
Response Type  Array
## Request Structure

Table 202: 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 that has this ID</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>delivery-service</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 that has this ID</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>
<tr>
<td>orderBy</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the nth page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</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.

### 370: Request Example

```
GET /api/1.4/origins?name=demo1 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
```
Response Structure

- **cachegroup**: A string that is the *name of the Cache Group* to which the *origin* belongs.
- **cachegroupId**: An integer that is the *ID of 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**: A string that is 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**: The *ID* of 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*.

# 371: 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=Mon, 18 Nov 2019
17:40:54 GMT; Max-Age=3600; HttpOnly

**Whole-Content-Sha512**: sm8DpvdvrfdSVLtmXTdfjsZbTlbc+pI40Gy0aj00X1URTPFfXuv/
4LgHb6A3r92iymbRhvFrH6qdB2g97U2sBg==

**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.

**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, integer which, if present, should be the Cache Group ID 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 ID of a Profile that shall be used by this origin
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

# 372: Request Example

```plaintext
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
```

¹ The tenantId field is required if and only if tenancy is enabled within Traffic Ops.
Content-Type: application/json

```json
{
    "deliveryServiceId": 2,
    "fqdn": "example.com",
    "name": "example",
    "port": 80,
    "protocol": "http",
    "tenantId": 1
}
```

Response Structure

- **cachegroup** A string that is the **name of the Cache Group** to which the **origin** belongs
- **cachegroupId** An integer that is the **ID of 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** The **ID** 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*.

### # 373: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: z4gp0MaqYu+gSRORhKT2eObVBUVDVx1rdteRaN5kRL9uJ3hNzUCi4dSKIt0rgNgOEDt6x/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,
        "tenantId": 1
    }
}
```
PUT

Updates an origin definition.

Auth. Required  Yes

Roles Required  “admin” or “operations”

Response Type  Object

Request Structure

Table 203: 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, integer which, if present, should be the Cache Group ID that identifies a Cache Group to which the new 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 ID of the Profile that shall be used by this origin</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 optional1, integral, unique identifier for the Tenant which shall own the new origin</td>
</tr>
</tbody>
</table>

# 374: Request Example

PUT /api/1.4/origins?id=2 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*

(continues on next page)
Cookie: mojolicious=...
Content-Length: 135
Content-Type: application/json

```json
{
    "deliveryServiceId": 2,
    "fqdn": "example.com",
    "isprimary": true,
    "name": "example",
    "port": 443,
    "protocol": "https",
    "tenantId": 1
}
```

**Response Structure**

- **cachegroup** A string that is the *name of the Cache Group* to which the *origin* belongs
- **cachegroupId** An integer that is the *ID of 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** The *ID* 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

# 375: 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: Zx7jo7A7QXr7DenYdvGQSoo0Pj4m0yYOAIeUpbdElmYMtDNYtPW82BCmMesFXkmP74nV4HbTUtYDHp
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>

#### # 376: Request Example

```
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

#### # 377: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: fLaY4/nh0yR38xq5weBKyg02+aQV6212roQ9UqUCHLMMrH1NMyhOHx+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 OS (Operating System) 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 of the Traffic Ops server’s Profile, if it is defined.

# 378: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: RxbRY2DZ+1YOdTzzUETEZ3wtLBiD2BwXMVuazjhe4a4cwgcZKRBWxZ6Oy5YYujFe1+UBiTG4sML/Amn27F4AVg==
Configuration File

The data returned from the endpoint comes directly from a configuration file. By default, the file is located at `/var/www/files/osversions.json`. The directory of the file can be changed by creating a specific Parameter named `kickstart.files.location` in configuration file `mkisofs`.

The format of the file is a JSON object as described in `Response Structure`.

```json
{
    "CentOS 7.2": "centos72"
}
```

# 379: Example osversions.json file

The legacy Perl Traffic Ops used a Perl configuration file located by default at `/var/www/files/osversions.cfg`. A Perl script is provided to convert the legacy configuration file to the new JSON format. The script is located within the Traffic Control repository at `traffic_ops/app/bin/osversions-convert.pl`.

```bash
./osversions-convert.pl < /var/www/files/osversions.cfg > /var/www/files/osversions.json
```

# 380: Example usage of conversion script

`parameterprofile`

**POST**

Create one or more `ParameterProfile` assignments.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Object
Request Structure

**paramId**  The *ID* of the *Parameter* to be assigned to the *Profiles* identified within the *profileIds* array

**profileIds**  An array of *Profile IDs* 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.

### # 381: 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,
    "profileIds": [18]
}
```

Response Structure

**paramId**  The *ID* of the *Parameter* which has been assigned to the *Profiles* identified within the *profileIds* array

**profileIds**  An array of *Profile IDs* 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.

### # 382: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: N2ahnhEnfZ0Uqny1N6Vu3HaOZk340YuiuyiqkhTbk0pENp+kwBPYu4Z/sgBAlc6xSO2aWlJzaeXw4u0B5heWz
```

(continues on next page)
parameters

GET

Gets all Parameters configured in Traffic Ops

Auth. Required  Yes
Roles Required  None
Response Type  Array
Request Structure

Table 205: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>configFile</td>
<td>no</td>
<td>Filter Parameters by Config File</td>
</tr>
<tr>
<td>id</td>
<td>no</td>
<td>Filters Parameters by ID</td>
</tr>
<tr>
<td>name</td>
<td>no</td>
<td>Filter Parameters by Name</td>
</tr>
<tr>
<td>orderBy</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit.</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n(^{th}) page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

# 383: 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 Parameter’s Config File
- **id** The Parameter’s ID
- **lastUpdated** The date and time at which this Parameter was last updated, in an ISO-like format
- **name** Name of the Parameter
- **profiles** An array of Profile Names that use this Parameter
- **secure** A boolean value that describes whether or not the Parameter is Secure
- **value** The Parameter’s Value
# 384: 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: UFO3/jcBFmFZM7CsrsIwTfPc5v8gUiXqJm6BNp1boPb4EQBnWNXZh/DbBwhMAOJeqDImoDlrLnrvjQG04AooA==
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 Issue #3093 for further developments

**configFile** The Parameter’s Config File

**name** Name of the Parameter

**secure** A boolean value that describes whether or not the Parameter is Secure

**value** The Parameter’s Value

### # 385: 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",
    "configFile": "records.config",
    "secure": false
}
```

### # 386: 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",
    "configFile": "records.config",
    "secure": false
},
{
    "name": "foo",
    "value": "bar",
    "configFile": "records.config",
    "secure": false
}
]```
Response Structure

- **configFile**: The Parameter’s Config File
- **id**: The Parameter’s ID
- **lastUpdated**: The date and time at which this Parameter was last updated, in an ISO-like format
- **name**: Name of the Parameter
- **profiles**: An array of Profile Names that use this Parameter
- **secure**: A boolean value that describes whether or not the Parameter is Secure
- **value**: The Parameter’s Value

# 387: Response Example - Single Object 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: eQrl48zWids0kDpfCYmmtYMpegjnFxfoVv1BYxxLSfp7P7p6oWX4uiC/+Cfh2X9i3G+M36eH95gukJqOB0GbQ==
X-Server-Name: traffic_ops_golang/
Date: Wed, 05 Dec 2018 19:18:21 GMT
Content-Length: 212

{
    "alerts": [
        {
            "text": "param was created.",
            "level": "success"
        }
    ],

    "response": {
        "configFile": "records.config",
        "id": 124,
        "lastUpdated": "2018-12-05 19:18:21+00",
        "name": "test",
        "profiles": null,
        "secure": false,
        "value": "quest"
    }
}
```
parameters/{{ID}}

GET

Gets details about a specific Parameter

Auth. Required  Yes
Roles Required  None
Response Type  Array

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 ID of the Parameter to be inspected</td>
</tr>
</tbody>
</table>

Table 207: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>orderBy</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>
<tr>
<td>sortOrder</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (&quot;desc&quot;)</td>
</tr>
<tr>
<td>limit</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>Return the n&lt;sup&gt;th&lt;/sup&gt; page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

# 388: 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 Parameter’s Config File
  id  The Parameter’s ID
lastUpdated  The date and time at which this Parameter was last updated, in an ISO-like format

name  Name of the Parameter

profiles  An array of Profile Names that use this Parameter

secure  A boolean value that describes whether or not the Parameter is Secure

value  The Parameter’s Value

# 389: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: UFO3/jcBFmFZM7csr1WfpC5v8gUixQqJm6BNp1boPb4E98BnwxZ/DbwHMAoJqeqDImoDlrLnrVjQG04AooA==
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
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 ID of the Parameter which will be deleted</td>
</tr>
</tbody>
</table>

- **configFile**  The Parameter’s Config File
- **name** Name of the Parameter
- **secure** A boolean value that describes whether or not the Parameter is Secure
- **value** The Parameter’s Value

# 390: Request Example

```plaintext
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

- **configFile**  The Parameter’s Config File
- **id** The Parameter’s ID
- **lastUpdated** The date and time at which this Parameter was last updated, in an ISO-like format
- **name** Name of the Parameter
- **profiles** An array of Profile Names that use this Parameter
- **secure** A boolean value that describes whether or not the Parameter is Secure
- **value** The Parameter’s Value
# 391: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: DMxS2gKceFVKRtezON/vsnrC+zI8onASSHaGv5i3wwvUvyt9KEe72gxQd6zgVcsSq3k8ZpkH6g3UI/WtEfdp5vA==
X-Server-Name: traffic_ops_golang/
Date: Wed, 05 Dec 2018 20:21:07 GMT
Content-Length: 209

```json
{
    "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 209: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The ID of the Parameter which will be deleted</td>
</tr>
</tbody>
</table>

# 392: Request Example

```
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

# 393: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: hJjQq2Seg7sqWt+jKgp6gwRxUtoVU34PFoc9wEaweXdaIBTn/BscoUuyw2/n+V8GZPqpeQcih2E50/0oQhdtHw==
X-Server-Name: traffic_ops_golang/
Date: Wed, 05 Dec 2018 19:20:30 GMT
Content-Length: 60

{ "alerts": [
    {
        "text": "param was deleted."
    },
    "level": "success"
]
```

parameters/{{ID}}/profiles

Deprecated since version ATCv4.
GET

Retrieves all Profiles assigned to a specific Parameter.

Auth. Required    Yes
Roles Required    None
Response Type    Array

Request Structure

Table 210: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The ID of the Parameter for which Profiles shall be listed</td>
</tr>
</tbody>
</table>

# 394: Request Structure

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  The Profile’s Description
id  The Profile’s ID
lastUpdated  The date and time at which this Profile was last updated, in an ISO-like format
name  The Profile’s Name
routingDisabled  The Profile’s Routing Disabled setting
type  The Profile’s Type

# 395: 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

(continues on next page)
parameters/{{ID}}/unassigned_profiles

Deprecated since version ATCv4.

**Warning:** There are very few good reasons to use this endpoint - be sure to limit said use.

**GET**

Retrieves all *Profiles* to which the specified *Parameter* is *not* assigned.

- **Auth. Required**  Yes
- **Roles Required**  None
- **Response Type**  Array
Request Structure

### Table 211: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The ID of the Parameter for which unassigned Profiles shall be listed</td>
</tr>
</tbody>
</table>

#### # 396: Request Example

```plaintext
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** The Profile’s Description
- **id** The Profile’s ID
- **lastUpdated** The date and time at which this Profile was last updated, in an ISO-like format
- **name** The Profile’s Name
- **routingDisabled** The Profile’s Routing Disabled setting
- **type** The Profile’s Type

#### # 397: 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, 05 Dec 2018 21:47:48 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: c669pfp2I0FB2xQ1v6RKDbu1d5cXvAtGnr7wEzW4ku+7EimNvEyBBPPW4G7FLRQvwO0y/OhWoJcm4/2YGBR39g==
Transfer-Encoding: chunked
```

(continues on next page)
{ "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"
    }
],
"alerts": [
    {
        "level": "warning",
        "text": "This endpoint is deprecated, please use 'GET /profiles' instead"
    }
]}

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}}

GET

Gets details about a specific Profile’s Parameters

Auth. Required Yes

Roles Required None

Response Type Array
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 Profile for which Parameters will be listed</td>
</tr>
</tbody>
</table>

# 398: 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 Parameter’s Config File
id The Parameter’s ID
lastUpdated The date and time at which this Parameter was last updated, in an ISO-like format
name Name of the Parameter
profiles An array of Profile Names that use this Parameter
secure A boolean value that describes whether or not the Parameter is Secure
value The Parameter’s Value

# 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: NudgZXUNyKNpmSFf856KEjyy+Pin/bFhG9NoRBDaxYbRkt2T5fP5e7sUNzfF15n/ZZsgbx6Tsgtfd7oM6j+eg==
X-Server-Name: traffic_ops_golang/
Date: Wed, 05 Dec 2018 21:08:56 GMT
Content-Length: 542

{ "response": [ (continues on next page) }
parameters/validate

Deprecated since version ATCv4.

**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**

- **configFile** The *Parameter’s Config File*
- **name** *Name of the Parameter*
- **secure** A boolean value that describes whether or not the *Parameter is Secure*
- **value** The *Parameter’s Value*

### # 400: Request Example

```json
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 *Parameter’s Config File*
- **id** The *Parameter’s ID*
- **name** *Name of the Parameter*
- **secure** A boolean value that describes whether or not the *Parameter is Secure*
- **value** The *Parameter’s Value*

### # 401: 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=...; Path=/; Expires=Mon, 18 Nov 2019
17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512:
CcsN9WhMPnv1PtBACtneCIlmleMLZxEywSmq3rdCclyyPu0cMgefRVI/aRYe+IDAKWmpesZHG+g1Ed1R7dfWg==
Content-Length: 149

{
  "alerts": [
    {
      "level": "success",
      "text": "Parameter exists."
    },
    {
      "level": "warning",
      "text": "This endpoint is deprecated, please use 'GET /parameters' instead"
    }
  ],
  "response": {
    "value": "bar",
    "name": "foo",
    "secure": 0,
    "id": 125,
    "configFile": "records.config"
  }
}

# 402: 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=...; Path=/; Expires=Mon, 18 Nov 2019
17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512:
kUNe70iQz1eEjsSZK3hk3WaJ3eTbsepDRUYeXTgEIII31BD5NiXobShT6zGhWJTsa18bNegjWbfAW
XEQQ==
Content-Length: 116
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 213: Request Query Parameters

<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>
<tr>
<td>orderBy</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>
<tr>
<td>sortOrder</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

# 403: Request Example

```
GET /api/1.4/phys_locations?name=CDN_in_a_Box 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
- **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

---

# 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: 0g4b3W1AwXytCnBo8TReQQij2v9oHA17MG9KuwMig5V4sFcMM5qP8dgPsFTunFr00DP120c7BpUbZs
X-Server-Name: traffic_ops_golang/
Date: Wed, 05 Dec 2018 22:19:52 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"
      }
   ]
}
```

**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 location
- **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

# 405: 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",
}
```

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.
"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
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

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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly

(continues on next page)
phys_locations/{{ID}}

GET

Retrieves information about a specific Physical Location

Auth. Required  Yes
Roles Required  None
Response Type  Array
Request Structure

Table 214: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>orderby</td>
<td>no</td>
<td>Choose the ordering of the results - must be the name of one of the fields of the response array</td>
</tr>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

Table 215: 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>

# 407: 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
- **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 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

# 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: 0g4b3W1AwXytCnBo8TReqQij2v9oHA17MG9KuwMi54sFcMM5qP8dgPsFTunFr00DPI20c7BpUbZs
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 216: 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>

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
e-mail  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 location\(^1\)
regionId  An integral, unique identifier for the region within which the physical location lies\(^1\)
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

# 409: Request Structure

```bash
PUT /api/1.4/phys_locations/2 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 268
Content-Type: application/json
```

\(^1\) 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.
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

# 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
(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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: qnMe60qxjSU8H1njlh00HWNr20YnV1OqfqCTdMBcdC1322jk2IFQsQQ3Xu0OR0WSh7OHCfXqDC1/jA1xjA==
X-Server-Name: traffic_ops_golang/
Date: Wed, 05 Dec 2018 23:39:17 GMT
Content-Length: 385

```json
{
  "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 217: 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>Physical Location</em> being deleted</td>
</tr>
</tbody>
</table>

# 411: 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

# 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: KeW/tEmICwpCQC8F0YMTqHdeR9J6W6Z3w/U+HOSbeCGyaEheCIhIsWlngT3dyfHltiu8UyzaPB6QrJyxGybBkw==
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"  
    }  
]  
```
Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

No parameters available

Response Structure

name  The name of the Physical Location

# 413: 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: s4/q6oyQHa+mQ3d3gRGhvVrsryvrsrkKxYnP574rVVUji0hHxYDboNYPPswi4MuwQRM7dZq8cp4/iw9rlLRkB0g==
X-Server-Name: traffic_ops_golang/
Date: Wed, 05 Dec 2018 22:35:02 GMT
Content-Length: 78

{
    "response": [
        {
            "name": "CDN_in_a_Box"
        },
        {
            "name": "Apachecon North America 2018"
        }
    ]
}
```

profileparameter

See also:
profileparameters.
POST

Create one or more Profile/Parameter assignments.

Auth. Required Yes
Roles Required “admin” or “operations”
Response Type Object

Request Structure

**profileId** The ID of the Profile to which the Parameters identified within the parameterIds array will be assigned

**paramIds** An array of Parameter IDs which shall be assigned to the Profile identified by profileId

**replace** An optional boolean (default: false) which, if true, will cause any conflicting Profile/Parameter assignments to be overridden.

# 414: Request Example

```json
{  "profileId": 18,  "paramIds": [2, 3]
}
```

Response Structure

**profileId** The ID of the Profile to which the Parameters identified within the parameterIds array are assigned

**paramIds** An array of Parameter IDs which have been assigned to the Profile identified by profileId

**replace** An optional boolean (default: false) which, if true, indicates that any conflicting Profile/Parameter assignments have been overridden.
# 415: 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=Mon, 18 Nov 2019_17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: N2ahnhEnf20Uqnjy1N6VuO2k340YuiuyiqkhTbk0pENp+kwBPYu4Z/sqBAloCfXSQaWlJzaeXw4uOD5heWw==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 15:18:23 GMT
Content-Length: 147

{
    "alerts": [
        {
            "text": "2 parameters were assigned to the 18 profile",
            "level": "success"
        }
    ],
    "response": {
        "profileId": 18,
        "paramIds": [
            2,
            3
        ],
        "replace": false
    }
}
```

profileparameters

GET

Retrieves all Parameter/Profile assignments.

Auth. Required  Yes

Roles Required  None

Response Type  Array
Request Structure

Table 218: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>orderBy</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

Response Structure

**lastUpdated** The date and time at which this Profile/Parameter association was last modified, in an ISO-like format

**parameter** The ID of a Parameter assigned to profile

**profile** The Name of the Profile to which the Parameter identified by parameter is assigned

# 416: Response Structure

```json
{
    "response": [
        {
            "lastUpdated": "2018-12-05 17:50:49+00",
            "profile": "GLOBAL"
        }
    ]
}
```

(continues on next page)
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 a Parameter to a 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 ID of a Parameter to assign to some Profile
- **profileId** The ID of the Profile to which the Parameter identified by parameterId will be assigned

# 417: 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
```
Array Format

Caution: Array format is broken as of the time of this writing. Follow Issue #3103 for further developments.

- **parameterId** The ID of a Parameter to assign to some Profile
- **profileId** The ID of the Profile to which the Parameter identified by parameterId will be assigned

# 418: Request Example - Array 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: 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 an ISO-like format
- **parameter** Name of the Parameter which is assigned to profile
- **parameterId** The ID of the assigned Parameter
- **profile** Name of the Profile to which the Parameter is assigned
profileId The ID of the Profile to which the Parameter identified by parameterId is assigned

# 419: Response Example - Single Object 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: eDmIwlzX44f2dxLRPHMNasGAK5qQv9Y70A2eeQHfEkliU4evwcsQ4WeHcH013/wPTGlpYoCgwLo8LQQpUxWQ==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 13:50:11 GMT
Content-Length: 166

{
  "alerts": [
    {
      "text": "profileParameter was created.",
      "level": "success"
    }
  ],
  "response": {
    "lastUpdated": null,
    "profile": null,
    "profileId": 18,
    "parameter": null,
    "parameterId": 1
  }
}
```

profileparameters/{{profileID}}/{{parameterID}}

DELETE

Deletes a Profile/Parameter association.

Auth. Required Yes

Roles Required "admin" or "operations"

Response Type undefined
## Request Structure

### Table 219: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>profileID</td>
<td>The ID of the Profile from which a Parameter shall be removed</td>
</tr>
<tr>
<td>parameterID</td>
<td>The ID of the Parameter which shall be removed from the Profile identified by profileID</td>
</tr>
</tbody>
</table>

### # 420: Request Example

```text
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

### # 421: Response Example

```text
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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512:JQuBqHyT9mNw09NSIDVQhkRtXdeAJc95WlpF2dwQeoBFmf0Y8knXm3/O/\rbJDEoUC7DhUQNlaoyIsqmqz4qQQ==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 15:00:15 GMT
Content-Length: 71

```json```
{
    "alerts": [
        {
            "text": "profileParameter was deleted.",
            "level": "success"
        }
    ]
}
```

**profiles**
GET

Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

Table 220: 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>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 ID</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 ID of a Parameter associated with them</td>
</tr>
</tbody>
</table>

# 422: Request Example

```shell
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 name of the CDN to which this Profile belongs

description  The Profile’s Description

id  The Profile’s ID

lastUpdated  The date and time at which this Profile was last updated, in an ISO-like format

name  The Profile’s Name

routingDisabled  The Profile’s Routing Disabled setting

type  The Profile’s Type
# 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: *
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: QEpKM/DwHBRvue9K7KrpwKFkWh6yCMQ2vSQgxE7dWFGJaqC4KOU92bsJU/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",
        "cdn": 2,
        "routingDisabled": false,
        "type": "ATS_PROFILE"
    }
]}
```

**POST**

Creates a new Profile.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Object

**Request Structure**

- **cdn** The integral, unique identifier of the CDN to which this Profile shall belong
- **description** The Profile’s Description
- **name** The Profile’s Name
- **routingDisabled** The Profile’s Routing Disabled setting
type  The Profile’s Type

# 424: Request Example

```
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 name of the CDN to which this Profile belongs
description  The Profile’s Description
id  The Profile’s ID
lastUpdated  The date and time at which this Profile was last updated, in an ISO-like format
name  The Profile’s Name
routingDisabled  The Profile’s Routing Disabled setting
type  The Profile’s Type

# 425: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: "UGV3PCnYBY0J3siICR0f9VVRNdUK1+9zsDDP6T9yt6t+AoHckHe6bvzOli9to/5fGH2zz519Nc1ro4taJUBD8g=="
```

(continues on next page)
profiles/{{ID}}

GET

**Auth. Required** Yes

**Roles Required** None

**Response Type** Array

**Request Structure**

### Table 221: Request Path Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The <em>ID</em> of the <em>Profile</em> to be retrieved</td>
</tr>
</tbody>
</table>

---

# 426: 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 name of the CDN to which this Profile belongs
- **description** The Profile’s Description
- **id** The Profile’s ID
- **lastUpdated** The date and time at which this Profile was last updated, in an ISO-like format
- **name** The Profile’s Name
- **params** An array of Parameters in use by this Profile
  - **configFile** The Parameter’s Config File
  - **id** The Parameter’s ID
  - **lastUpdated** The date and time at which this Parameter was last updated, in an ISO-like format
  - **name** Name of the Parameter
  - **profiles** An array of Profile Names that use this Parameter
  - **secure** A boolean value that describes whether or not the Parameter is Secure
  - **value** The Parameter’s Value
- **routingDisabled** The Profile’s Routing Disabled setting
- **type** The Profile’s Type

# 427: 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=/; Expires=Mon, 18 Nov 2019 ...
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 request payload

Auth. Required Yes

Roles Required “admin” or “operations”

Response Type Object
Request Structure

Table 222: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The ID of the Profile being modified</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdn</td>
<td>The integral, unique identifier of the CDN to which this Profile will belong</td>
</tr>
<tr>
<td>description</td>
<td>The Profile’s new Description</td>
</tr>
<tr>
<td>name</td>
<td>The Profile’s new Name</td>
</tr>
<tr>
<td>routingDisabled</td>
<td>The Profile’s new Routing Disabled setting</td>
</tr>
<tr>
<td>type</td>
<td>The Profile’s new Type</td>
</tr>
</tbody>
</table>

Warning: Changing this will likely break something, be VERY careful when modifying this value

# 428: 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdn</td>
<td>The integral, unique identifier of the CDN to which this Profile belongs</td>
</tr>
<tr>
<td>cdnName</td>
<td>The name of the CDN to which this Profile belongs</td>
</tr>
<tr>
<td>description</td>
<td>The Profile’s Description</td>
</tr>
<tr>
<td>id</td>
<td>The Profile’s ID</td>
</tr>
<tr>
<td>lastUpdated</td>
<td>The date and time at which this Profile was last updated, in an ISO-like format</td>
</tr>
</tbody>
</table>
name  The Profile’s Name
routingDisabled  The Profile’s Routing Disabled setting
type  The Profile’s Type

# 429: Response Example

```json
{
    "alerts": [
        {
            "text": "profile was updated.",
            "level": "success"
        }
    ],
    "response": {
        "id": 16,
        "lastUpdated": "2018-12-07 21:45:06+00",
        "name": "test",
        "description": "A test profile for API examples",
        "cdnName": null,
        "cdn": 2,
        "routingDisabled": true,
        "type": "UNK_PROFILE"
    }
}
```

DELETE

Allows user to delete a Profile.

Auth. Required  Yes

Roles Required  “admin” or “operations”

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 ID of the Profile being deleted</td>
</tr>
</tbody>
</table>

# 430: 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

# 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: HNmJkZaNW9yil08/3TnqZ5Fl1H6Rptjgp3KI46F2d0jLYcu+8jEhDLl1okoirdrHyU4R1c3hjCI0urN7PVVWDA==
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}}/export

GET

Auth. Required  Yes

5.1. Traffic Ops API
Roles Required  None
Response Type  Object

Request Structure

Table 224: Request Path Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the Profile to be exported</td>
</tr>
</tbody>
</table>

# 432: Request Example

GET /api/1.1/profiles/3/export  HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...

Response Structure

profile  The exported Profile
        cdn  The name of the CDN to which this Profile belongs
        description  The Profile’s Description
        name  The Profile’s Name
        type  The Profile’s Type

parameters  An array of Parameters in use by this Profile
        config_file  The Parameter’s Config File
        name  Name of the Parameter
        value  The Parameter’s Value

# 433: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Set-Mime-Version, Cookie
Access-Control-Allow-Methods: POST,GET,OPTIONS,PUT,DELETE
Access-Control-Allow-Origin: *
Content-Disposition: attachment; filename="GLOBAL.json"
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
profiles/{{ID}}/parameters

GET

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>The ID of the Profile for which Parameters will be listed</td>
</tr>
</tbody>
</table>

5.1. Traffic Ops API
# 434: 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 Parameter’s Config File
- **id** The Parameter’s ID
- **lastUpdated** The date and time at which this Parameter was last updated, in an ISO-like format
- **name** Name of the Parameter
- **profiles** An array of Profile Names that use this Parameter
- **secure** A boolean value that describes whether or not the Parameter is Secure
- **value** The Parameter’s Value

# 435: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: NudgZXUNyKNpmSFf856KEjyy+Pin/bFhG9NoRBDAxYbRk7t5fF52e7sUN2fF15n/2Zsgbx6Tsgtd7oM6j+eg==
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,
        "value": "Traffic Ops CDN"
    }
]}
```
POST

Associates Parameters to a Profile. If the Parameter does not exist, creates it and associates it to the Profile. If the Parameter already exists, associates 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 226: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The ID of 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 *Parameter’s* *Config File*
- **name**  *Name* of the *Parameter*
- **secure**  A boolean value that describes whether or not the *Parameter* is *Secure*
- **value**  The *Parameter’s* *Value*

#### # 436: Response Example - Single Parameter Format

```
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 *Parameter’s* *Config File*
- **name**  *Name* of the *Parameter*
- **secure**  A boolean value that describes whether or not the *Parameter* is *Secure*
- **value**  The *Parameter’s* *Value*
# 437: 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 Parameter’s Config File
  - **name** Name of the Parameter
  - **secure** A boolean value that describes whether or not the Parameter is Secure
  - **value** The Parameter’s Value
- **profileId** The ID of the Profile to which the Parameter(s) have been assigned
- **profileName** Name of the Profile to which the Parameter(s) have been assigned

# 438: Response Example - Single Parameter 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
```

(continues on next page)
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 227: Request Path Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The ID of the Profile for which unassigned Parameters will be listed</td>
</tr>
</tbody>
</table>

#### # 439: Request Example

```plaintext
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 Parameter’s Config File
- **id** The Parameter’s ID
- **lastUpdated** The date and time at which this Parameter was last updated, in an ISO-like format
- **name** Name of the Parameter
- **profiles** An array of Profile Names that use this Parameter
- **secure** A boolean value that describes whether or not the Parameter is Secure
- **value** The Parameter’s Value

#### # 440: 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512:iO7YHU+0spCPSaR6cDrVIQwxSS1GoSyi8K6ng4eemuxqOxB9FdfPgpXN8w+хmxf22wRMLXHv5S6cf...
X-Server-Name: traffic_ops_golang/
Date: Wed, 05 Dec 2018 21:37:50 GMT
```
Transfer-Encoding: chunked

{
  "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/import**

**POST**

Imports a Profile that was exported via `profiles/[[ID]]/export`

*Note:* On import of the Profile Parameters if a Parameter already exists with the same Name, Config File and Value it will link that to the Profile instead of creating it.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Object

**Request Structure**

**profile** The exported Profile

**cdn** The name of the CDN to which this Profile belongs
description The Profile’s Description
name The Profile’s Name
type The Profile’s Type
parameters An array of Parameters in use by this Profile
config_file The Parameter’s Config File
name Name of the Parameter
value The Parameter’s Value

# 441: Request Example

POST /api/1.1/profiles/import HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.62.0
Accept: */*
Cookie: mojolicious=...
Content-Type: application/json

```json
{
  "profile": {
    "name": "GLOBAL",
    "description": "Global Traffic Ops profile",
    "cdn": "ALL",
    "type": "UNK_PROFILE"
  },
  "parameters": [
    {
      "config_file": "global",
      "name": "tm.instance_name",
      "value": "Traffic Ops CDN"
    },
    {
      "config_file": "global",
      "name": "tm.toolname",
      "value": "Traffic Ops"
    }
  ]
}
```

Response Structure

cdn The name of the CDN to which this Profile belongs
description The Profile’s Description
name The Profile’s Name
type The Profile’s Type
id The Profile’s ID

5.1. Traffic Ops API
# 442: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512:
X-Server-Name: traffic_ops_golang/
Date: Fri, 13 Sep 2019 20:14:42 GMT
Transfer-Encoding: gzip

{
  "alerts": [
    {
      "level": "success",
      "text": "Profile imported [ Global ] with 2 new and 0 existing parameters"
    }
  ],
  "response": {
    "cdn": "ALL",
    "name": "Global",
    "id": 18,
    "type": "UNK_PROFILE",
    "description": "Global Traffic Ops profile"
  }
}
```

```
profiles/name/{{name}}/copy/{{copy}}

POST

Copy Profile to a new Profile. The new Profile's Name must not already exist.

Auth. Required  Yes

Roles Required  “admin” or “operations”

Response Type  Object
```
Request Structure

Table 228: 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>

# 443: Request Example

```plaintext
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 new Profile’s Description
- **id**  The ID of the new Profile
- **idCopyFrom**  The ID of 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

# 444: 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: Fri, 07 Dec 2018 22:03:54 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: r6V9viEZu1WCns0AUGEx1MttxjXiU8SZV0tSqjeq72JDLl5s8fMmjJdR/HRWdUhHn7Ax6GzYhoKwntJjMyc7ZWg==
Content-Length: 252
```

{ "alerts": [ (continues on next page) }
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 229: 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>

# 445: 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 Parameter’s Config File
id The Parameter’s ID
lastUpdated The date and time at which this Parameter was last updated, in an ISO-like format

name Name of the Parameter

profiles An array of Profile Names that use this Parameter

secure A boolean value that describes whether or not the Parameter is Secure

value The Parameter’s Value

# 446: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: NudgZXUNyKNpmSFf85bKEjyy+Pin/„bFhG9NoRBDaxYbRkt2T5F52e?sunZfF15n/Z2sgbx6Tsgtfd7oM6j+eg==
X-Server-Name: traffic_ops_golang/
Date: Wed, 05 Dec 2018 21:52:08 GMT
Content-Length: 542

{ "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",
      "secure": false,
      "value": "1"
    }
]
POST

Associates Parameters to a Profile. If the Parameter does not exist, creates it and associates it to the Profile. If the Parameter already exists, associates 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 230: 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 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 Parameter’s Config File
- **name**  Name of the Parameter
**secure** A boolean value that describes whether or not the *Parameter* is *Secure*

**value** The *Parameter’s Value*

# 447: Request Example - Single Parameter Format

```plaintext
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 *Parameter’s Config File*

**name** *Name* of the *Parameter*

**secure** A boolean value that describes whether or not the *Parameter* is *Secure*

**value** The *Parameter’s Value*

# 448: Request Example - Parameter Array Format

```plaintext
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
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",
    }
]
```

(continues on next page)
Response Structure

**parameters**  An array of objects representing the *Parameters* which have been assigned

- **configFile**  The Parameter’s *Config File*
- **name**  *Name* of the Parameter
- **secure**  A boolean value that describes whether or not the Parameter is *Secure*
- **value**  The Parameter’s *Value*

**profileId**  The ID of the *Profile* to which the Parameter(s) have been assigned

**profileName**  *Name* of the *Profile* to which the Parameter(s) have been assigned

# 449: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: HQWqOkoYHjvcxheWFrHOboZnUC+qLG1L04ojtsLNnZyVUIu/qgJrzzpKq3FEHUWaZrnDCZM/12D8AXOKBw==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 14:20:22 GMT
Content-Length: 357

{
  "alerts": [
    {
      "text": "Assign parameters successfully to profile test",
      "level": "success"
    }
  ],
  "response": {
    "parameters": [
      {
        "configFile": "quest",
        "secure": 0
      }
    ]
  }
}
```


```json
"name": "test",
"secure": 0,
"value": "A test parameter for API examples",
"id": 126,
}
{
  "configFile": "bar",
  "name": "foo",
  "secure": 0,
  "value": "Another test parameter for API examples",
  "id": 129
}
}

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 re- sponse (represents the contents of the requested configuration file)

Request Structure

Table 231: 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 ID 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>

5.1. Traffic Ops API
# 450: 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.

# 451: 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 15:23:44 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019
  → 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: C1Hrs4y3qSThOeZJo5aDJu1QjD/r/
  → 7vO6c7E7TaFXx67kWat9luk9BSvieXN5yrOE4HkGs1GBkNzj23hb5mYw==
Content-Length: 211

# DO NOT EDIT - Generated for ATS_MID_TIER_CACHE by Traffic Ops
  → (trafficops.infra.ciab.test:443) on Thu Nov 15 15:23:44 UTC 2018
# TRAFFIC OPS NOTE: This is running with forced volumes - the size
  → is irrelevant
volume=1 scheme=http size=100%
```

```
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

# 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: 1XiReeWNZfrLjdordjrZJxJSleAJ8v3rOulOnmBXEfTe+Sn3cKx3Pa0Rch4TII4ck/93sI+5LI1m6MvTCaQ==
X-Server-Name: traffic_ops_golang/
Date: Fri, 07 Dec 2018 20:51:28 GMT
Content-Length: 360

{  "response" : [ 
{  "name" : "GLOBAL" },
{  "name" : "TRAFFIC_ANALYTICS" },
{  "name" : "TRAFFIC_OPS" },
{  "name" : "TRAFFIC_OPS_DB" },
{  "name" : "TRAFFIC_PORTAL" },
{  "name" : "TRAFFIC_STATS" },
{  "name" : "INFLUXDB" },
{  "name" : "RIAK_ALL" },
{  "name" : "ATS_EDGE_TIER_CACHE" },
{  "name" : "ATS_MID_TIER_CACHE" },
{  "name" : "BIND_ALL" },
{  "name" : "CCR_CIAB" },
{  "name" : "ENROLLER_ALL" },
{  "name" : "RASCAL-Traffic_Monitor" }
]}
```
regions

GET

Retrieves information about Regions

Auth. Required Yes
Roles Required None
Response Type Object

Request Structure

Table 232: 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>
<tr>
<td>orderBy</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the nth page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

# 453: 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

---

### # 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: nSYbR+fRXaxhYl7dWgf0Udo2AsiXEnvvED1CPbk7ZNWK03I3TOhtmCQx9AbnJJ6xKYnlIt6EKMeopVT
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\(^\d\)
- **divisionName** The name of the division which shall contain the new region\(^\d\)
- **name** The name of the region

# 455: 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

# 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
```

\(^\d\) 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).
DELETE

New in version 1.5.

Deletes a region. If no query parameter is specified, a 400 Bad Request response is returned.

Auth. Required Yes

Roles Required “admin” or “operations”

Response Type Object

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>Delete Region by integral, unique identifier</td>
</tr>
<tr>
<td>name</td>
<td>no</td>
<td>Delete Region by name</td>
</tr>
</tbody>
</table>
# 457: Request Example

```plaintext
DELETE /api/1.5/regions?name=Manchester HTTP/1.1
User-Agent: curl/7.29.0
Host: trafficops.infra.ciab.test
Accept: */*
Cookie: mojolicious=...
```

Response Structure

# 458: 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=/; Expires=Fri, 07 Feb 2020 13:56:24 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: yNqXKcoiohEbjReKh8LDzifh87dIusuvUqgQnYueyKqCXkfd5bQvQ0hQ2AA0S2a/oe2SAqMjojGsUlxHtIQw==
X-Server-Name: traffic_ops_golang/
Date: Fri, 07 Feb 2020 12:56:24 GMT
Content-Length: 62

```json
{
    "alerts": [
        {
            "text": "region was deleted.",
            "level": "success"
        }
    ]
}
```

`regions/{{ID}}`

GET

Retrieves a specific Region.

**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 region to inspect</td>
</tr>
</tbody>
</table>

# 459: 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>

# 460: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: nSYbR+fRXaxhYl7dWgf0Udo2AsiXEnwvED1CPbk7ZNK03I3T0htmCQx9ABnJ6xKYnlt6EKMeopVTV
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",
    }
]}

(continues on next page)
PUT

Updates a Region.

Auth. Required Yes

Role(s) Required “admin” or “operator”

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 region to update</td>
</tr>
</tbody>
</table>
| division | The new integral, unique identifier of the division which shall contain the region
| divisionName | The new name of the division which shall contain the region
| name  | The new name of the region                        |

# 461: Request Example

```
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"
}
```

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).
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

# 462: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: 7SVj4q7dtSTNQEJlBApEwlad28WBVFnpdHaatoIpNfeLltfcpcdVTcOKB4JXQv7r1SD2p/TxBQc6ExpwYtKnK==
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 ATCv4.

5.1. Traffic Ops API  717
POST

Creates a new Physical Location within the specified region.

Auth. Required  Yes

Roles Required  “admin” or “operations”

Response Type  Object

Request Structure

Table 236: 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>

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",
    "name": "London",
    "address": "1 Buckingham Palace",
    "city": "London",
    "name": "London"
}

(continues on next page)
"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

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

# 464: 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
### roles

**GET**

Retrieves all user *Roles*.

- **Auth. Required** Yes
- **Roles Required** None
- **Response Type** Array
Request Structure

Table 237: 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 Role identified by this integral, unique identifier</td>
</tr>
<tr>
<td>name</td>
<td>no</td>
<td>Return only the Role with this name</td>
</tr>
<tr>
<td>privLevel</td>
<td>no</td>
<td>Return only those Roles that have this privilege level</td>
</tr>
<tr>
<td>orderBy</td>
<td>no</td>
<td>Choose the ordering of the results - must be the name of one of the fields of the response array</td>
</tr>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the nth page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

# 465: Request Example

GET /api/1.4/roles?name=admin HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...

Response Structure

- **capabilities** An array of the names of the Capabilities given to this Role
  
  New in version 1.3.

- **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

# 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

(continues on next page)
POST

New in version 1.3.

Creates a new *Role*.

**Auth. Required** Yes

**Roles Required** “admin”

**Response Type** Object

**Request Structure**

- **capabilities** An optional array of capability names that will be granted to the new *Role*

- **description** A helpful description of the *Role*’s purpose.

- **name** The name of the new *Role*

- **privLevel** The privilege level of the new *Role*¹

---

¹ *privLevel* cannot exceed the privilege level of the requesting user. Which, of course, must be the privilege level of “admin”. Basically, this means that there can never exist a *Role* with a higher privilege level than “admin”.

---

722 Chapter 5. APIs
# 467: Request Example

```
POST /api/1.3/roles HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 56
Content-Type: application/json

{
    "name": "test",
    "description": "quest",
    "privLevel": 30
}
```

## Response Structure

**capabilities**  An array of the names of the Capabilities given to this *Role*

**Tip:** This can be null or empty, depending on whether it was present in the request body, or merely empty. Obviously, it can also be a populated array.

**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*

# 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: gzfc7m/in5vWsVP+y9h6JfDhpXKn9VAzoIpENhKbQfP8Q6jug08Rt2AK/3Nz1cx5z8P9IjVxDdIg7m1C8bw==
X-Server-Name: traffic_ops_golang/
Date: Wed, 04 Sep 2019 17:44:42 GMT
```

(continues on next page)
PUT

New in version 1.3.

Replaces an existing Role with one provided by the request.

**Auth. Required** Yes

**Roles Required** “admin”

**Response Type**

**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 Role to be replaced</td>
</tr>
</tbody>
</table>

**Warning:** When not present, the affected Role’s Capabilities will be unchanged - not removed, unlike when the array is empty.

capabilities  An optional array of capability names that will be granted to the new Role

description  A helpful description of the Role’s purpose.

name  The new name of the Role

privLevel  The new privilege level of the new Role
# 469: Request Example

```
PUT /api/1.3/roles?id=5 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 56
Content-Type: application/json

{
    "name":"test",
    "privLevel": 29,
    "description": "quest"
}
```

**Response Structure**

**capabilities** An array of the names of the Capabilities given to this *Role*

_Tip:_ This can be null or empty, depending on whether it was present in the request body, or merely empty. Obviously, it can also be a populated array.

**Warning:** If no **capabilities** array was given in the request, this will always be null, even if the *Role* has Capabilities that would have gone unchanged.

**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*

# 470: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
```
DELETE

New in version 1.3.

Deletes a Role

**Auth. Required** Yes

**Roles Required** “admin”

**Response Type** undefined

Request Structure

Table 239: 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 Role to be replaced</td>
</tr>
</tbody>
</table>

# 471: Request Example

```
DELETE /api/1.3/roles?id=5 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```
Response Structure

# 472: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: 10jeFZihtbvAus/XyHAW8rhgS9JBD+X/ezCp1iExYEcHxN4gjr1L6x8zDFXORueBS1F1dgtxbWKT7QsmwCHUWA==
X-Server-Name: traffic_ops_golang/
Date: Thu, 05 Sep 2019 13:02:06 GMT
Content-Length: 59

{
  "alerts": [
    {
      "text": "role was deleted.",
      "level": "success"
    }
  ]
}
```

server_capabilities

New in version 1.4.

GET

Retrieves Server Capabilities.

- **Auth. Required**: Yes
- **Roles Required**: “read-only”
- **Response Type**: Array

Request Structure

Table 240: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>no</td>
<td>Return the Server Capability with this name</td>
</tr>
</tbody>
</table>
# 473: Request Structure

```
GET /api/1.4/server_capabilities?name=RAM HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

## Response Structure

- **name**: The name of this *Server Capability*
- **lastUpdated**: The date and time at which this *Server Capability* was last updated, in ISO-like format

# 474: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: EH8jo8OrCu79Tz9xpgT3YRyKJ/p2NcTmbS3huwtgRByHz9H6q2LQja59RlPaVSq3ZxsU6QhTa0x5nBkQ9LPSAA==
X-Server-Name: traffic_ops_golang/
Date: Mon, 07 Oct 2019 21:36:13 GMT
Content-Length: 68

```json
{
    "response": [
        {
            "name": "RAM",
            "lastUpdated": "2019-10-07 20:38:24+00"
        }
    ]
}
```

**POST**

Create a new *Server Capability*.

- **Auth. Required**: Yes
- **Roles Required**: “admin” or “operations”
Response Type Object

Request Structure

name The name of the Server Capability

# 475: Request Example

<table>
<thead>
<tr>
<th>POST</th>
<th>/api/1.4/server_capabilities</th>
<th>HTTP/1.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>trafficops.infra.ciab.test</td>
<td></td>
</tr>
<tr>
<td>User-Agent</td>
<td>curl/7.47.0</td>
<td></td>
</tr>
<tr>
<td>Accept</td>
<td><em>/</em></td>
<td></td>
</tr>
<tr>
<td>Cookie</td>
<td>mojolicious=...</td>
<td></td>
</tr>
<tr>
<td>Content-Length</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Content-Type</td>
<td>application/json</td>
<td></td>
</tr>
</tbody>
</table>

{
    "name": "RAM"
}

Response Structure

name The name of this Server Capability

lastUpdated The date and time at which this Server Capability was last updated, in ISO-like format

# 476: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: ysdopC//JQI79BRUa61s6M2HzHxYHpo5RdcuauOqCYxiVOoUhNZfOVydVkv8zDN2qA374XKnym4kWj3VzQlXg
X-Server-Name: traffic_ops_golang/
Date: Mon, 07 Oct 2019 22:10:00 GMT
Content-Length: 137

{
    "alerts": [
        {
            "text": "server capability was created."
        }
    ]
}
DELETE

Deletes a specific Server Capability.

Auth. Required  Yes
Roles Required  “admin” or “operations”
Response Type  undefined

Request Structure

Table 241: Request Query 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 Server Capability to be deleted</td>
</tr>
</tbody>
</table>

# 477: Request Example

DELETE /api/1.4/server_capabilities?name=RAM HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...

Response Structure

# 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
server_server_capabilities

New in version 1.4.

GET

Gets all associations of Server Capabilities to Cache Servers.

Auth. Required  Yes
Roles Required  None
Response Type  Array
## Request Structure

Table 242: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverId</td>
<td>no</td>
<td>Filter Server Capability associations by the integral, unique identifier of the server to which they are assigned</td>
</tr>
<tr>
<td>server-Host-Name</td>
<td>no</td>
<td>Filter Server Capability associations by the host name of the server to which they are assigned</td>
</tr>
<tr>
<td>server-Capability</td>
<td>no</td>
<td>Filter Server Capability associations by Server Capability name</td>
</tr>
<tr>
<td>orderBy</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit.</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n(^{th}) page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

### # 479: Request Example

```
GET /api/1.4/server_server_capabilities HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

## Response Structure

- **serverHostName** The server’s host name
- **serverId** The server’s integral, unique identifier
- **lastUpdated** The date and time at which this association between the server and the Server Capability was last updated, in an ISO-like format
- **serverCapability** The Server Capability’s name
# 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: UFO3/jcBFmFZM7CsrsIwTfPc5v8gUixXqJm6BNp1boPb4EQBnWNXZh/
X-Server-Name: traffic_ops_golang/
Date: Mon, 07 Oct 2019 22:15:11 GMT
Content-Length: 150

{
    "response": [
        {
            "lastUpdated": "2019-10-07 22:05:31+00",
            "serverHostName": "atlanta-org-1",
            "serverId": 260,
            "serverCapability": "ram"
        },
        {
            "lastUpdated": "2019-10-07 22:05:31+00",
            "serverHostName": "atlanta-org-2",
            "serverId": 261,
            "serverCapability": "disk"
        }
    ]
}
```

POST

 Associates a Server Capability to a server.

 **Auth. Required**  Yes

 **Roles Required**  “admin” or “operations”

 **Response Type**  Object

**Request Structure**

 serverId  The integral, unique identifier of a server to be associated with a Server Capability
serverCapability  The *Server Capability*’s name to associate

**Note:** The server referenced must be either an *Edge-tier* or *Mid-tier Cache Server*.

### # 481: Request Example

```plaintext
POST /api/1.4/server_server_capabilities HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 84
Content-Type: application/json

{
    "serverId": 1,
    "serverCapability": "disk"
}
```

### Response Structure

**serverId**  The integral, unique identifier of the newly associated server

**lastUpdated**  The date and time at which this association between the server and the *Server Capability* was last updated, in an ISO-like format

**serverCapability**  The *Server Capability*’s name

### # 482: 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=/; Expires=Mon, 18 Nov 2019
    17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: eQrl48zWidsOkDpfCYmmtYMpegjnFx0Vv1BYxLSfp7P7p6oWX4uiC/+Cfh2X9i3G+MQ36eH95gukJqOB0GboQ==
X-Server-Name: traffic_ops_golang/
Date: Mon, 07 Oct 2019 22:15:11 GMT
Content-Length: 157

{
    "alerts": [
```

(continues on next page)
DELETE

Disassociate a server from a Server Capability.

Note: If the serverCapability is a Server Capability required by a Delivery Service that to which the server is assigned the DELETE will be blocked until either the server is unassigned from the Delivery Service or the Server Capability is no longer required by the Delivery Service.

Auth. Required  Yes

Roles Required  “admin” or “operations”

Response Type  undefined

Request Structure

Table 243: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverId</td>
<td>yes</td>
<td>The integral, unique identifier of the server to disassociate</td>
</tr>
<tr>
<td>serverCapability</td>
<td>yes</td>
<td>term:Server Capability name to disassociate from given server</td>
</tr>
</tbody>
</table>

# 483: Request Example

DELETE /api/1.4/server_server_capabilities?serverId=1&
        serverCapability=disk HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Response Structure

# 484: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: UFO3/jcBFmFZM7CsrsIwTfPc5v8gUixQjm6BNp1boPb4EQBnWNXZh/
X-Server-Name: traffic_ops_golang/
Date: Mon, 07 Oct 2019 22:15:11 GMT
Content-Length: 96

{
    "alerts": [
        {
            "text": "server server_capability was deleted.",
            "level": "success"
        }
    ]
}
```

**servercheck**

Updates the resulting value from running a given check extension on a server.

**POST**

Post a server check result to the “serverchecks” table.

Auth. Required Yes

Roles Required None

---

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.
Response Type  Object

Request Structure

The request only requires to have either host_name or id defined.

**host_name** The hostname of the server to which this “servercheck” refers.

**id** The id of the server to which this “servercheck” refers.

**servercheck_short_name** The short name of the “servercheck”.

**value** The value of the “servercheck”

# 485: Request Example

```json
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": 1
}
```

Response Structure

# 486: Response Example

```json
{
    "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 244: 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 that has this ID</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 that has this ID</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n th page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

# 487: 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**  A string that is the name of the Cache Group to which the server belongs
- **cachegroupId**  An integer that is the ID of the Cache Group to which the 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 FQDN

guid An identifier used to uniquely identify the server

Note: 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 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 MTU 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 OF-FLINE status

---

1 For more information see the Wikipedia page on Lights-Out management.
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 used by this server

profileDesc  A Description of the Profile used by this server

profileId  The ID 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

# 488: Response Example

HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true

(continues on next page)
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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: WyapQctUIhjzEALka5QbBiZKZ58M1c6MJSwjBeGyJS2UzbL3W61N/4kvA2tPrP4qMWQBWz6JjbF7Y51NRASUmQ==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 16:13:31 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,
        "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 An integer that is the ID of the Cache Group to which the 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 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 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 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 An optional string containing the user name for the server’s ILO service
interfaceMtu The MTU 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
**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** The ID the Profile that shall be 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

---

# 489: Request Example

```json
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**: A string that is the *name of the Cache Group* to which the server belongs
- **cachegroupId**: An integer that is the *ID of the Cache Group* to which the 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 FQDN
- **guid**: An identifier used to uniquely identify the server

**Note**: 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 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 server’s ILO service user - displays as **simply ****** if the currently logged-in user does not have the ‘admin’ or ‘operations’ ROLE(S) <ROLE>
- **iloUsername**: The user name for the server’s ILO service
- **interfaceMtu**: The MTU 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 used by this server

profileDesc  A Description of the Profile used by this server

profileId  The ID 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

# 490: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: mcGmmu5ONDg3jmvlkItcw6jxiT1ecmePVuj2fmKizrn5ThKjsSadeJIynaeOK0XVUjHuYHdtdynSx
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",
    "ipAddress": "0.0.0.1",
    "ipGateway": "0.0.0.2",
    "ipNetmask": "255.255.255.0",
    "lastUpdated": "2018-12-10 17:44:04+00",
    ...
  }
}
servers/checks

Deals with the various checks associated with certain types of servers.

See also:

*Check Extensions*

**GET**

Fetches identifying and meta information as well as “check” values regarding all servers that have a *Type* with a name beginning with “EDGE” or “MID” (ostensibly this is equivalent to all *cache servers*).

**Auth. Required**  Yes

**Roles Required**  None

**Response Type**  Array

**Request Structure**

No parameters available.
Response Structure

**adminState** The name of the server’s Status - called “adminState” for legacy reasons

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

**checks** An optionally present map of the names of “checks” to their values. Only numeric and boolean checks are represented, and boolean checks are represented as integers with 0 meaning “false” and 1 meaning “true”. Will not appear if the server in question has no valued “checks”.

**hostName** The (short) hostname of the server

**id** The server’s integral, unique identifier

**profile** The name of the Profile used by the server

**revalPending** A boolean that indicates whether or not the server has pending revalidations

**type** The name of the server’s Type

**updPending** A boolean that indicates whether or not the server has pending updates

# 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-Encoding: gzip
Content-Type: application/json
Set-Cookie: mojolicious=...; Path=/; Expires=Thu, 23 Jan 2020 20:00:19 GMT; Max-Age=3600; HttpOnly
X-Server-Name: traffic_ops_golang/
Date: Thu, 23 Jan 2020 19:00:19 GMT
Content-Length: 202

{
  "response": [
    {
      "adminState": "REPORTED",
      "cacheGroup": "CDN_in_a_Box_Edge",
      "id": 12,
      "hostName": "edge",
      "revalPending": false,
      "profile": "ATS_EDGE_TIER_CACHE",
      "type": "EDGE",
      "updPending": false
    }
  ]
}
```
servers/hostname/{{name}}/details

GET

Retrieves the details of a server.

Auth. Required Yes

Roles Required None

Response Type Object

Request Structure

Table 245: 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** A string that is the *name of the Cache Group* to which the 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 FQDN

**guid** An identifier used to uniquely identify the server
<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostName</td>
<td>The (short) hostname of the server</td>
</tr>
<tr>
<td>httpsPort</td>
<td>The port on which the server listens for incoming HTTPS connections/requests</td>
</tr>
<tr>
<td>id</td>
<td>An integral, unique identifier for this server</td>
</tr>
<tr>
<td>iloIpAddress</td>
<td>The IPv4 address of the server’s ILO service</td>
</tr>
<tr>
<td>iloIpGateway</td>
<td>The IPv4 gateway address of the server’s ILO service</td>
</tr>
<tr>
<td>iloIpNetmask</td>
<td>The IPv4 subnet mask of the server’s ILO service</td>
</tr>
<tr>
<td>iloPassword</td>
<td>The password of the server’s ILO service user</td>
</tr>
<tr>
<td>iloUsername</td>
<td>The user name for the server’s ILO service</td>
</tr>
<tr>
<td>interfaceMtu</td>
<td>The MTU to configured on interfaceName</td>
</tr>
<tr>
<td>interfaceName</td>
<td>The name of the primary network interface used by the server</td>
</tr>
<tr>
<td>ip6Address</td>
<td>The IPv6 address and subnet mask of interfaceName</td>
</tr>
<tr>
<td>ip6Gateway</td>
<td>The IPv6 address of the gateway used by interfaceName</td>
</tr>
<tr>
<td>ipAddress</td>
<td>The IPv4 address of interfaceName</td>
</tr>
<tr>
<td>ipGateway</td>
<td>The IPv4 address of the gateway used by interfaceName</td>
</tr>
<tr>
<td>ipNetmask</td>
<td>The IPv4 subnet mask used by interfaceName</td>
</tr>
<tr>
<td>offlineReason</td>
<td>A user-entered reason why the server is in ADMIN_DOWN or OFFLINE status</td>
</tr>
<tr>
<td>physLocation</td>
<td>The name of the physical location where the server resides</td>
</tr>
<tr>
<td>profile</td>
<td>The Name of the Profile used by this server</td>
</tr>
<tr>
<td>profileDesc</td>
<td>A Description of the Profile used by this server</td>
</tr>
<tr>
<td>rack</td>
<td>A string indicating “server rack” location</td>
</tr>
<tr>
<td>routerHostName</td>
<td>The human-readable name of the router responsible for reaching this server</td>
</tr>
<tr>
<td>routerPortName</td>
<td>The human-readable name of the port used by the router responsible for reaching this server</td>
</tr>
<tr>
<td>status</td>
<td>The status of the server</td>
</tr>
</tbody>
</table>

See also:

1 For more information see the Wikipedia page on Lights-Out management.
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

# 492: 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=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: ZDeQrG0D7Q3Wy3ZEUT9t21QQ9F9Yc3RR/QR91n22Un1YubdhdkI1r3B+LYP5ZKkVg8ByrVFPyx6Nac0iBTGTQ==
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": "",
  "rack": "",
  "mgmtIpGateway": "",
  "deliveryservices": [
```
(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 246: 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>

# 493: Request Example

```
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
- **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

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.

**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.
Apache Traffic Control Documentation, Release 4.0.0

no reason for this field to ever be false.

# 494: 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: 4R6bjNVrcecHGN3eGDgQ1yDiBnEDGQe7QtOMIsRw1pck9SZR8chRQznkTF3YdROAZ118BxR3fXTlKkQfjf0jvlK6OoKdA==
X-Server-Name: traffic_ops_golang/
Date: Mon, 04 Feb 2019 16:24:01 GMT
Content-Length: 174

[{
    "host_name": "edge",
    "upd_pending": false,
    "revalPending": false,
    "use_reval_pending": true,
    "host_id": 10,
    "status": "REPORTED",
    "parent_pending": false,
    "parent_reval_pending": false
}]
```

**servers/{{ID}}**

**GET**

Retrieves properties of a specific server.

**Auth. Required** Yes

**Roles Required** None

**Response Type** Array

5.1. Traffic Ops API 755
Request Structure

Table 247: 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>

Table 248: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>orderBy</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the nᵗʰ page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

# 495: 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** A string that is the *name of the Cache Group* to which the server belongs

**cachegroupId** An integer that is the *ID of the Cache Group* to which the 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

---

**Note:** 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 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 used by this server

---

1 For more information see the Wikipedia page on Lights-Out management.
**profileDesc**  A *Description* of the *Profile* used by this server

**profileId**  The *ID* 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

---

# 496: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
```
```json
{
  "response": [
    {
      "cacheGroupId": 6,
      "cdnId": 2,
      "cdnName": "CDN-in-a-Box",
      "domainName": "infra.ciab.test",
      "id": 10,
      "interfaceMtu": 1500,
      "interfaceName": "eth0",
      "ipAddress": "172.16.239.120",
      "ipGateway": "172.16.239.1",
      "ipNetmask": "255.255.255.0",
      "lastUpdated": "2018-12-05 18:45:05+00",
      "profile": "ATS_MID_TIER_CACHE",
      "profileDesc": "Mid Cache - Apache Traffic Server",
      "rack": "",
      "revalPending": false,
      "status": "REPORTED",
      "statusId": 3,
      "tcpPort": 80,
      "type": "MID",
      "typeId": 12,
      "iloIpAddress": "",
      "iloIpGateway": "",
      "iloIpNetmask": "",
      "iloPassword": "",
      "iloUsername": "",
      "mgmtIpAddress": "",
      "mgmtIpGateway": "",
      "mgmtIpNetmask": "",
      "offlineReason": "",
      "physLocation": "Apachecon North America 2018",
      "physLocationId": 1,
      "ip6Address": "fc01:9400:1000:8::120",
      "ip6Gateway": "fc01:9400:1000:8::1",
      "ip6Netmask": "fc01:9400:1000:8::1",
      "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,
    }
  ]
}```
PUT

Allow user to edit a server.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Object

### Request Structure

Table 249: 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>

**cachegroupId** An integer that is the *ID of the Cache Group* to which the 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 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 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 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) <ROLE>

**iloUsername** An optional string containing the user name for the server’s ILO service

**interfaceMtu** The MTU 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
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 The ID the Profile that shall be 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.

---

**# 497: 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
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**: A string that is the *name of the Cache Group* to which the server belongs.

- **cachegroupId**: An integer that is the *ID of the Cache Group* to which the 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 FQDN.

- **guid**: An identifier used to uniquely identify the server.

  **Note:** 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 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 used by this server

profileDesc A Description of the Profile used by this server

profileId The ID 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

# 498: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: 9lGAMCCC9I/bOpuBSyf3ACffjHeRuXCuxrA/oU78uYzW5PeFTq5PHSSnsnqKG5EOvWg0Rko0CwguGeNc9IT0w==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 17:58:57 GMT
Content-Length: 848

{
  "alerts": [
    {
      "text": "server was updated."
    }
  ],
  "response": {
    "cachegroupId": 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",
  }
}
```
 DELETE

Allow user to delete server through api.

Auth. Required Yes

Roles Required "admin" or "operations"

Response Type undefined

Request Structure

Table 250: 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>

# 499: Request Example

DELETE /api/1.4/servers/13 HTTP/1.1
Host: trafficops.infra.ciab.test
Response Structure

# 500: 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: JZdjKJYWN9w9NF6VE/rVkGqecycKB2AbkkI4LnDmgpJLwu53bRHAA+4uWrow0zuba/4MSEhHKshutiypSxpge= 
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 18:23:21 GMT
Content-Length: 61
```

```json
{
   "alerts": [
      {
         "text": "server was deleted.",
         "level": "success"
      }
   ]
}
```

**servers/{{ID}}/deliveryservices**

**GET**

Retrieves all Delivery Services assigned to a specific server.

- **Auth. Required**: Yes
- **Roles Required**: None
- **Response Type**: Array

1 Only the Delivery Services visible to the requesting user’s Tenant will appear, regardless of their Role or the Delivery Services’ actual ‘server assignment’ status.
Request Structure

Table 251: 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 for which assigned Delivery Services shall be listed</td>
</tr>
</tbody>
</table>

Table 252: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>order</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the nth page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

# 501: Request Example

GET /api/1.4/servers/9/deliveryservices HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...

Response Structure

active A boolean that defines Active.
anonymous BlockingEnabled A boolean that defines Anonymous Blocking
cacheurl A Cache URL Expression

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 DNS TTL - 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 A Check Path
consistentHashRegex A Consistent Hashing Regular Expression
   New in version 1.4.
consistentHashQueryParams An array of Consistent Hashing Query Parameters
   New in version 1.4.
depCachingType The Deep Caching setting for this Delivery Service
   New in version 1.3.
displayName The Display Name
dnsBypassCname A DNS Bypass CNAME
dnsBypassIp A DNS Bypass IP
dnsBypassIp6 A DNS Bypass IPv6
dnsBypassTtl The DNS Bypass TTL
dscp A DSCP to be used within the Delivery Service
ecsEnabled A boolean that defines the EDNS0 Client Subnet Enabled setting on this Delivery Service
   New in version 1.4.
edgeHeaderRewrite A set of Edge Header Rewrite Rules
eexampleURLs An array of Example URLs
fqPacingRate The Fair-Queuing Pacing Rate Bps
   New in version 1.3.
geoLimit An integer that defines the Geo Limit
geoLimitCountries A string containing a comma-separated list defining the Geo Limit Countries
geoLimitRedirectUrl A Geo Limit Redirect URL
depProvider The Geolocation Provider
globalMaxMbps The Global Max Mbps
globalMaxTps The Global Max TPS
httpBypassFqdn A HTTP Bypass FQDN
id An integral, unique identifier for this Delivery Service
infoUrl An Info URL
initialDispersion The Initial Dispersion
ipv6RoutingEnabled A boolean that defines the IPv6 Routing Enabled setting on this Delivery Service
lastUpdated  The date and time at which this Delivery Service was last updated, in RFC 3339 format

logsEnabled  A boolean that defines the Logs Enabled setting on this Delivery Service

longDesc  The Long Description of this Delivery Service

longDesc1  The Long Description 2 of this Delivery Service

longDesc2  The Long Description 3 of this Delivery Service

matchList  The Delivery Service’s Match List

  pattern  A regular expression - the use of this pattern is dependent on the type field (backslashes are escaped)

  setNumber  An integer that provides explicit ordering of Match List items - this is used as a priority ranking by Traffic Router, and is not guaranteed to correspond to the ordering of items in the array.

  type  The type of match performed using pattern.

maxDnsAnswers  The Max DNS Answers allowed for this Delivery Service

maxOriginConnections  The Max Origin Connections

  New in version 1.4.

midHeaderRewrite  A set of Mid Header Rewrite Rules

missLat  The Geo Miss Default Latitude used by this Delivery Service

missLong  The Geo Miss Default Longitude used by this Delivery Service

multiSiteOrigin  A boolean that defines the use of Use Multi-Site Origin Feature by this Delivery Service

orgServerFqdn  The Origin Server Base URL

originShield  A Origin Shield string

profileDescription  The Description of the Profile with which this Delivery Service is associated

profileId  The ID of the Profile with which this Delivery Service is associated

profileName  The Name of the Profile with which this Delivery Service is associated

protocol  An integral, unique identifier that corresponds to the Protocol used by this Delivery Service

qstringIgnore  An integral, unique identifier that corresponds to the Query String Handling setting on this Delivery Service

rangeRequestHandling  An integral, unique identifier that corresponds to the Range Request Handling setting on this Delivery Service
regexRemap  A *Regex Remap Expression*

regionalGeoBlocking  A boolean defining the *Regional Geoblocking* setting on this *Delivery Service*

remapText  *Raw Remap Text*

signed  true if and only if signingAlgorithm is not null, false otherwise

signingAlgorithm  Either a *Signing Algorithm* or null to indicate URL/URI signing is not implemented on this *Delivery Service*

New in version 1.3.

sslKeyVersion  This integer indicates the *SSL Key Version*

tenantId  The integral, unique identifier of the *Tenant* who owns this *Delivery Service*

New in version 1.3.

trRequestHeaders  If defined, this defines the *Traffic Router Log Request Headers* used by Traffic Router for this *Delivery Service*

New in version 1.3.

trResponseHeaders  If defined, this defines the *Traffic Router Additional Response Headers* used by Traffic Router for this *Delivery Service*

New in version 1.3.

type  The *Type* of this *Delivery Service*

typeld  The integral, unique identifier of the *Type* of this *Delivery Service*

xmlId  This *Delivery Service*’s *xml_id*

---

# 502: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: CFmtW41aoDezCYxtAXnS54dfFOD6jdxDj2/LMpbBqnnny5kac7JQhdfFAWF109s195XVSUV85JHFzX2Tw/mJabQ==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Jun 2019 17:01:30 GMT
Content-Length: 1500

{  "response": [   
```
"active": true,
"anonymousBlockingEnabled": false,
"cacheurl": null,
"ccrDnsTtl": null,
"cdnId": 2,
"cdnName": "CDN-in-a-Box",
"checkPath": null,
"displayName": "Demo 1",
"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": "2019-06-10 15:14:29+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": 2,
"queryStringIgnore": 0,
servers/{{ID}}/queue_update

**Caution:** In the vast majority of cases, it is advisable that the PUT method of the `servers/{{ID}}` endpoint be used instead.

**POST**

**Queue** or dequeue updates for a specific server.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Object
## Request Structure

Table 253: 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** Queue Updates for the server, propagating configuration changes to the actual server
- **dequeue** Cancels any pending updates on the server

### # 503: 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** Queue Updates was performed on 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

### # 504: 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
```

(continues on next page)
servers/{{ID}}/status

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 254: 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 |

# 505: Request Example

```
PUT /api/1.4/servers/13/status HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```
Response Structure

# 506: Response Example

```json
{
    "status": "ADMIN_DOWN",
    "offlineReason": "Bad drives"
}
```

**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 255: 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>
</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
  - **profileName** The Name of the Profile used by this server
  - **profileId** The ID the Profile used by this 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

---

**5.1. Traffic Ops API**

---

777
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

# 507: 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 15:28:10 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512:  
X-K6pRI4MkN8O9+wKW8MG3w6nTnmLHtC2KqzXCjw4JfoMYIVJC6fVTN9ysGML71VF27ZAIP1TveWhja
fN7sQ==
Transfer-Encoding: chunked

{ "info": {
    "profileId": 9,
    "toUrl": null,
    "serverIpv4": "172.16.239.100",
    "serverTcpPort": 80,
...
```

1 Exactly one of these fields is guaranteed to exist for any given configuration file - although “apiUrl” is far more common.
"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"
},
{
    "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"
},

(continues on next page)
{
    "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"
},
{
    "fnameOnDisk": "remap.config",
    "location": "/etc/trafficserver/",
    "apiUri": "/api/1.2/servers/edge/configfiles/ats/remap.config",
    "scope": "servers"
},
{
    "fnameOnDisk": "storage.config",
    "location": "/etc/trafficserver/",
    "apiUri": "/api/1.2/profiles/ATS_EDGE_TIER_CACHE/configfiles/ats/storage.config",
    "scope": "profiles"
},
(continues on next page)
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 256: 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>

# 508: 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
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.

---

# 509: 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 15:32:25 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: EmhHogPfcxQq2zHmFFJtjwzZiUHNgOZvE572Se/H/54gwarkKjm89+xJr7fQbfytc7xWYApzwfjN16LfbM0hg==
Content-Length: 107
```

---

**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

Table 257: 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 status counts for only servers of this Type</td>
</tr>
</tbody>
</table>

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.

# 510: Response Example

```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
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=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: M072YRXvtNwijnCfntv/\nW3AsSpOhC17Cpm0UDznOcXxwwgRYSGXx2MoeeovXSNzYim62FJJoQJom1ccRSAW92McA==
Content-Length: 38

{ "response": {
    "REPORTED": 2,
    "ONLINE": 9
}}
```

**servers/totals**

Deprecated since version 1.1.
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

# 511: Response Example

```json
{
    "alerts": [
        {
            "level": "warning",
            "text": "This endpoint is deprecated"
        }
    ],
    "response": [
        {
            "count": 1,
            "type": "EDGE"
        }
    ]
}
```

(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*.

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** undefined

### Request Structure

Table 258: 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 <em>Snapshot</em> shall be taken</td>
</tr>
</tbody>
</table>

#### # 512: 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

#### # 513: 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: z4PhNX7vuL3xVChQ1m2AB9Yg5AULVxXcg/SpIdNs6c5H0NE8XYysP+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
```
**GET**

Retrieve all static DNS entries configured within Traffic Control

- **Auth. Required**: Yes
- **Roles Required**: None
- **Response Type**: Array

**Request Structure**

Table 259: 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 the Cache Group that has this Name</td>
</tr>
<tr>
<td>cachegroupId</td>
<td>no</td>
<td>Return only static DNS entries assigned to the Cache Group that has this ID</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>
# 514: Request Example

```
GET /api/1.4/staticdnsentries?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, integer that is the *ID 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 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.
# 515: Response Example

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: Px1zTH3ihg+hfmADGcap0Juud39fGsx5Y3CzqaFmRwFu1ZLMzOsy0EN2pb7vp0tp16/zeIUYAC3dbsBwOmA==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 20:04:33 GMT
Content-Length: 226

{
    "response": [
        {
            "address": "foo.bar",
            "cachegroup": null,
            "cachegroupId": null,
            "deliveryservice": "demo1",
            "deliveryserviceId": 1,
            "host": "test",
            "id": 2,
            "lastUpdated": "2018-12-10 19:59:56+00",
            "ttl": 300,
            "type": "CNAME_RECORD",
            "typeId": 40
        }
    ]
}
```

POST

New in version 1.3.

Creates a new, static DNS entry.

- **Auth. Required** Yes
- **Roles Required** “admin” or “operations”
- **Response Type** Object

5.1. Traffic Ops API
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, integer that is the **ID 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 FQDN which shall resolve to `address`.

- **ttl**  The TTL of this static DNS entry in seconds.

- **typeId**  The integral, unique identifier of the **Type** of this static DNS entry.

---

# 516: Request Example

```json
POST /api/1.4/staticdnentries HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 92
Content-Type: application/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, integer that is the ID 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 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

# 517: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: 8dcJyjw2NJZx0L9Oz16P7g/7j5AljlpyiY6Y+rRVQ2wGcwYI3yiGPPrz6ur0qKzqgEBBsh8aPF44WTHAR9jUJdg==
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.

**Auth. Required** Yes

**Role(s) Required** "admin" or "operator"

**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 the static DNS entry to modify</td>
</tr>
</tbody>
</table>

**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, integer that is the *ID of a Cache Group* which will service this static DNS entry

**deliveryserviceId** The integral, unique identifier of a *Delivery Service* under the domain of which this static DNS entry shall be active

**Note:** This field has no effect, and is not used by any part of Traffic Control. It exists for legacy compatibility reasons.
**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 TTL of this static DNS entry in seconds.

**typeId** The integral, unique identifier of the *Type* of this static DNS entry.

### # 518: Request Example

```bash
PUT /api/1.4/staticdnsentries?id=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, integer that is the *ID 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 FQDN which shall resolve to address

id  An integral, unique identifier for this static DNS entry

ttl  The 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

# 519: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: +FaYmpnlIIzVSBq0nosw29N2cV9x9hVgWuUqXUyiDihVUSzX4jrdAloRDgzDvKsYQB8LSkPdGHwt1
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 19:59:56 GMT
Content-Length: 279

{
  "alerts": [
    {
      "text": "staticDNSEntry was updated.",
      "level": "success"
    }
  ],
  "response": {
    "address": "foo.bar",
    "cachegroup": null,
    "cachegroupId": null,
    "deliveryservice": null,
    "deliveryserviceId": 1,
    "host": "test",
    "id": 2,
    "lastUpdated": "2018-12-10 19:59:56+00",
    "ttl": 300,
    "type": null,
    "typeId": 40
  }
}
```
DELETE

New in version 1.3.
Delete staticdnsentries.

Auth. Required Yes
Roles Required “admin” or “operations”
Response Type undefined

Request Structure

Table 261: 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>

# 520: 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

# 521: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: g6uqHPU44LuTtqU2ahtazrVCpcpNWVc9kJWQYRuiVLDnsm39KOB/xt3XM6jO/
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 20:05:52 GMT
Content-Length: 69

{ "alerts": [ (continues on next page)
Either retrieve a list of summary stats or the timestamp of the latest recorded stats summary. What is returned is driven by the query parameter `lastSummaryDate`.

If the parameter is set it will return an object with the latest timestamp, else an array of summary stats will be returned.

**Auth. Required** Yes

**Roles Required** None

**Response Type** Array or Object

### Request Structure
Summary Stats

Table 262: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>deliveryServiceName</td>
<td>no</td>
<td>Return only summary stats that were reported for <em>Delivery Service</em> with the given display name</td>
</tr>
<tr>
<td>cdnName</td>
<td>no</td>
<td>Return only summary stats that were reported for CDN with the given name</td>
</tr>
<tr>
<td>statName</td>
<td>no</td>
<td>Return only summary stats that were reported for given stat name</td>
</tr>
<tr>
<td>orderBy</td>
<td>no</td>
<td>Choose the ordering of the results - can only be one of deliveryServiceName, statName or cdnName</td>
</tr>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (”desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

# 522: Request Example

```
GET /api/1.4/stats_summary HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

Last Updated Summary Stat

Table 263: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>statName</td>
<td>no</td>
<td>Get lastest updated date for the given stat</td>
</tr>
<tr>
<td>lastSummaryDate</td>
<td>yes</td>
<td>Tells route to get only lastest updated timestamp</td>
</tr>
</tbody>
</table>

# 523: Request Example

```
GET /api/1.4/stats_summary?lastSummaryDate=true HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
```

(continues on next page)
Response Structure

Summary Stats

cdnName  The CDN name for which the summary stat was taken for

Note:  If the cdn is equal to all it represents summary_stats across all delivery services across all CDNs

deliveryServiceName  The Delivery Service display name for which the summary stat was taken for

Note:  If the deliveryServiceName is equal to all it represents summary_stats across all delivery services within the given CDN

statName  Stat name summary stat represents

statValue  Summary stat value

summaryTime  Timestamp of summary, in an ISO-like format

statDate  Date stat was taken, in RFC 3339 format

# 524: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: dHNip9kpTGGS1w39/fWcFehNktgmXZus8XaufnmDpv0PyG/3fK/KfoCO3Z0j9V74/CCffps7doEygWeL/xRtKA==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 20:56:59 GMT
Content-Length: 150

{ "response": [ (continues on next page)
Last Updated Summary Stat

**summaryTime** Timestamp of the last updated summary, in **RFC 3339** format

# 525: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: dHNip9kpTGG1w39/fWcFehNktgXZus8XaufmDpvoPyG/3fK/KfoCO3Zo9V74/CCffps7doEygWeL/xRtKA==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 20:56:59 GMT
Content-Length: 150
```
POST

New in version 1.5.

Post a stats summary for a given stat.

Auth. Required  Yes
Roles Required  None
Response Type  Object

Request Structure

cdnName  The CDN name for which the summary stat was taken for

Note: If the cdn is equal to all it represents summary_stats across all
delivery services across all CDNs

deliveryServiceName  The Delivery Service display name for which the summary
stat was taken for

Note: If the deliveryServiceName is equal to all it represents sum-
mary_stats across all delivery services within the given CDN

statName  Stat name summary stat represents
statValue  Summary stat value
summaryTime  Timestamp of summary, in an ISO-like format
statDate  Date stat was taken, in RFC 3339 format

Note: statName, statValue and summaryTime are required. If cdnName and
deliveryServiceName are not given they will default to all.

# 526: Request Example

POST  /api/1.5/stats_summary HTTP/1.1
Host:  trafficops.infra.ciab.test

{  "response": {        "summaryTime": "2019-11-19 00:04:06+00"  }}
Response Structure

# 527: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: ezxk+iP7o7KE7zpWmGc0j8nz5k+1wAzY0H1NiA2xswTQrt+N+6CgQqUV2r9G1HAsPNr0HF2PhYs/Xr7DrYOW0A==
X-Server-Name: traffic_ops_golang/
Date: Thu, 06 Dec 2018 02:14:45 GMT
Content-Length: 97

{ "alerts": [ {
    "text": "Stats Summary was successfully created",
    "level": "success"
} ]
}
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 <em>Statuses</em> with this <em>exact</em> description</td>
</tr>
<tr>
<td>id</td>
<td>no</td>
<td>Return only the <em>Status</em> with this integral, unique identifier</td>
</tr>
<tr>
<td>name</td>
<td>no</td>
<td>Return only <em>Statuses</em> with this name</td>
</tr>
<tr>
<td>orderby</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

# 528: Request Example

```
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 an ISO-like format
**name**  The name of the status

### # 529: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: dHNip9kpTGGSlw39/fWcFehNktgmxZus8XaufmDpv0PyG/3fK/KfoCO32Oj9V74/CCffps7doEygWel/xRtKA==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 20:56:59 GMT
Content-Length: 150

{  "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}}**

**GET**

Retrieves information about a particular *Status*

- **Auth. Required**  Yes
- **Roles Required**  None
- **Response Type**  Array

**Request Structure**

Table 265: 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>Status</em> being inspected</td>
</tr>
</tbody>
</table>
Table 266: Request Query Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>orderBy</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

# 530: Request Example

```
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 an ISO-like format
- **name** The name of the status

# 531: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: dHNip9kpTGG5iw39/+fWcFehNktgmX2us8Xaufnmdpv0PyG/3fK/KfoCO3Z0j9V74/CCffps7doEygWeL/xRtKA==
X-Server-Name: traffic_ops_golang/
```
steering/{{ID}}/targets

GET

Get all targets for a steering Delivery Service.

Auth. Required Yes

Roles Required None

Response Type Array

Request Structure

Table 267: 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 268: 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>
<tr>
<td>order</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>
<tr>
<td>sortOrder</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter. pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

# 532: 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
Accept: */*
Cookie: mojolicious=...
```

Response Structure

- **deliveryService** A string that is the `xml_id` of the steering Delivery Service
- **deliveryServiceId** An integral, unique identifier for the steering Delivery Service
- **target** A string that is 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 as an integer

# 533: 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)```
**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 Delivery Service 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

5.1. Traffic Ops API
# 534: Request Example

```plaintext
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**  A string that is the `xml_id` of the steering Delivery Service
- **deliveryServiceId**  An integral, unique identifier for the steering Delivery Service
- **target**  A string that is 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 as an integer

# 535: 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=;/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: +dTvfzrnOhdwAOMmY28r0+gFV5z+3aABI2FfAMziTYcU+pZrDanrJzMXpKWIL5Q/+oCUBZpJDrt9hRCFkT4oGYw==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Dec 2018 21:22:17 GMT
Content-Length: 196

(continues on next page)
{ "alerts": [
    {
      "text": "steeringtarget was created.",
      "level": "success"
    }
  ],
  "response": {
    "deliveryService": "test",
    "deliveryServiceId": 2,
    "target": "demol",
    "targetId": 1,
    "type": "HTTP",
    "typeId": 1,
    "value": 100
  }
}

steering/{{ID}}/targets/{{targetID}}

**GET**

Get a single target for a specific STEERING-Type Delivery Service.

**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 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>
<tr>
<td>orderBy</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>
<tr>
<td>sortOrder</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>
# 536: 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**  A string that is the `xml_id` of the steering *Delivery Service*
- **deliveryServiceId**  An integral, unique identifier for the steering *Delivery Service*
- **target**  A string that is 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 as an integer

# 537: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: utlJK4oYS2l6Ff7NzAqRuQeMEtazYn3rM3Nlux2XgTLxvSys1Hy0mJrwDExSU05gVMdrgYCLZrZEvP
X-Server-Name: traffic_ops_golang/
Date: Tue, 11 Dec 2018 14:16:53 GMT
Content-Length: 130

{ "response": [
   {
      "deliveryService": "test",
      "deliveryServiceId": 2,
      "target": "demo1",
      "targetId": 1,
      "type": "HTTP",
      "typeId": 1,
   }
]
```
"value": 100
}
]

**PUT**

Updates a steering target.

**Auth. Required** Yes

**Roles Required** Portal, Steering, Federation, “operations” or “admin”

**Response Type** Object

**Request Structure**

Table 271: 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>
<tr>
<td>typeId</td>
<td>The integral, unique identifier of the routing type of the target Delivery Service</td>
</tr>
<tr>
<td>value</td>
<td>The ‘weight’ which shall be attributed to the target Delivery Service</td>
</tr>
</tbody>
</table>

# 538: 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
Content-Type: application/json

{
    "value": 1,
    "typeId": 1
}
```

**Response Structure**

**deliveryService** A string that is the xml_id of the steering Delivery Service
deliveryServiceId An integral, unique identifier for the steering Delivery Service
target A string that is 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
typeld An integral, unique identifier for the routing type of this target Delivery Service
value The ‘weight’ attributed to this steering target as an integer

# 539: 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: AfXs1RzdtU3HZYkr93qBMVTZRJ5oTF2u5sKYnd+DSqz2+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": "demo1",
"targetId": 1,
"type": "HTTP",
"typeld": 1,
"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 272: 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>

# 540: 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

# 541: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: N6h8K17uQveqgTc3fmxKXY2yYe5smApNcaTow4ab0DHGFdJfQh89I4nvvaXvmVNHxVAgX3UE/6b1bO8/9Xqg==
X-Server-Name: traffic_ops_golang/
Date: Tue, 11 Dec 2018 14:42:54 GMT
Content-Length: 69

{ "alerts": [
  {
    "text": "steeringtarget was deleted.",
    "level": "success"
  }
]
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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: ObxOXk1jrCI/
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",
      "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**

---

## 5.1. Traffic Ops API
GET

Get all requested Tenants.

**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>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>
<tr>
<td>orderBy</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>
<tr>
<td>sortOrder</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>page</td>
<td>Return the nth page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page</td>
</tr>
</tbody>
</table>

# 543: 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
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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: Yzr6TfhxgpZ3pbbrr4TRG4wC3P1nHDDzs2igtz/1ppLSyz2MzugqaGW4y5yzwz15T3+7q6HWej7GQ2t1XlVezQ==
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

---

5.1. Traffic Ops API
**parentId** The integral, unique identifier of the parent of this tenant

# 545: Request Example

```plaintext
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

# 546: 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=/; Expires=Mon, 18 Nov 2019
17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: ysdopC//...
X-Server-Name: traffic_ops_golang/
Date: Tue, 11 Dec 2018 19:37:16 GMT
Content-Length: 162

{
    "alerts": [
        {
            "text": "tenant was created."
        }
    ]
}
```

(continues on next page)
{
  "response": {
    "id": 9,
    "name": "test",
    "active": true,
    "lastUpdated": "2018-12-11 19:37:16+00",
    "parentId": 1
  }
}

**tenants/{{ID}}**

**GET**

Get a specific *Tenant*.

**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 tenant being inspected</td>
</tr>
</tbody>
</table>

```plaintext
GET /api/1.4/tenants/1 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
# 548: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: Yzr6TfhxgpZ3pbbrr4TRG4wC3P1nHDDzgs2igtz/1ppLSy2MzugqaGW4y5yzwl5T3+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](https://github.com).  

Updates a specific tenant.

- **Auth. Required** Yes
- **Roles Required** “admin” or “operations”
- **Response Type** Object
Request Structure

Table 275: 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

# 549: Request Example

```
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",
   "parentId": 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

# 550: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
```

(continues on next page)
DELETE

**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>
<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>
# 551: Request Example

```plaintext
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

# 552: 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: KU0XIbFoD0Cy06kzH2Gl59pBqie/TEFJgh33msGnWxJ21RkTLaSTHT8Df4X+pOs7UauZH10akGvaOvUTiN/vg==
X-Server-Name: traffic_ops_golang/
Date: Tue, 11 Dec 2018 20:40:31 GMT
Content-Length: 61

```json```
{
  "alerts": [
    {
      "text": "tenant was deleted.",
      "level": "success"
    }
  ]
}
```
Apache Traffic Control Documentation, Release 4.0.0

Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

Table 277: 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 TO Extensions by the integral, unique identifier of an Extension</td>
</tr>
<tr>
<td>name</td>
<td>no</td>
<td>Filter TO Extensions by the name of an Extension</td>
</tr>
<tr>
<td>script_file</td>
<td>no</td>
<td>Filter TO Extensions by the base filename of the script that runs for the Extension</td>
</tr>
<tr>
<td>is-active</td>
<td>no</td>
<td>Boolean used to return either only active (1) or inactive(0) TO Extensions</td>
</tr>
<tr>
<td>type</td>
<td>no</td>
<td>Filter TO Extensions by the type of Extension</td>
</tr>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>Choose the maximum number of results to return</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit.</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the nth page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

# 553: Request Example

GET /api/1.5/to_extensions HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...

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
id  An integral, unique identifier for this extension definition

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

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 Check Type of the extension.

Changed in version 1.5: Since to_extensions returns configured TO Plugins in 1.5, this type will either be the stored extension type or TO_PLUGIN for TO Plugins.

version  A (hopefully) semantic version number describing the version of the plugin

# 554: 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=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: n73jg9XR4V5Cwqq56Rf3wuIi99k3mM5u2NAjcZ/-gQBu8jvAFymDln2qKeJ+wT111vjIsHpXCOVXV7+5UGakLgA==
Transfer-Encoding: chunked

{ "response": [ { "script_file": "ToPingCheck.pl", "version": "1.0.0" }, ...
```
(continues on next page)
"name": "ILO_PING",
"description": "null",
"info_url": "-",
"additional_config_json": "{ check_name: "ILO", 
"base_url": "https://localhost", "select": "ilo_ip_address",
"cron": "9 * * * *" },
"isactive": 1,
"type": "CHECK_EXTENSION_BOOL",
"id": 1,
"servercheck_short_name": "ILO"
},
{
"script_file": "ToPingCheck.pl",
"version": "1.0.0",
"name": "10G_PING",
"description": "null",
"info_url": "-",
"additional_config_json": "{ check_name: "10G", 
"base_url": "https://localhost", "select": "ip_address",
"cron": "18 * * * *" },
"isactive": 1,
"type": "CHECK_EXTENSION_BOOL",
"id": 2,
"servercheck_short_name": "10G"
}]

POST

Changed in version 1.5: Only supports CHECK_EXTENSION extensions now. Previous implementation would attempt to accept CONFIG_EXTENSION or STATISTIC_EXTENSION extensions but would fail the creation.

Creates a new Traffic Ops extension.

**Auth. Required**  Yes

**Roles Required**  None¹

**Response Type**  undefined

**Request Structure**

- `additional_config_json`: An optional string containing a JSON-encoded object with extra configuration options... inside a JSON object...

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.
**description**  A short description of the extension

**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

Changed in version 1.5: Prior to version 1.5, `isactive` could be given as a string or an integer. Now it can only be given as an integer.

**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.

   Changed in version 1.5: `type` now only accepts a CHECK_EXTENSION type with the naming convention of `CHECK_EXTENSION_*`.

**version**  A (hopefully) semantic version number describing the version of the plugin

   # 555: Request Example

```
POST /api/1.5/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",

(continues on next page)
```
"servercheck_short_name": "test",
"type": "CHECK_EXTENSION_NUM"
}

Response Structure

# 556: 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=...; Path=/; Expires=Mon, 18 Nov 2019
17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: 7M67PYn1i6WzGQFS3g8GhlSOyg6VENZMqm/
kUffOTLLPfuWSEuSLA65R5R+VyJ1NdqOG5Bp78mk+jYcqhVGw==
Content-Length: 89

{ "supplemental":
  { "id": 5
  },
  "alerts": [{
    "level": "success",
    "text": "Check Extension Loaded."
  }]
}

to_extensions/{{ID}}

New in version 1.5.

DELETE

Deletes a Traffic Ops extension definition. This does not delete the actual extension file.

Auth. Required Yes
Roles Required  None
Response Type  undefined

Request Structure

Table 278: 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>

# 557: Request Example

```
DELETE /api/1.5/to_extensions/16 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
```

Response Structure

# 558: 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 16:33:52 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019
17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512:
\nEBONu85azbGzaehDTA0DP3NPqWbByIza1XQhgwtsW2WTXyK/
dxtncp0YiJXyO0tH9H+n+6BBfojBOb5h0dFPA==
Content-Length: 60

{ "alerts": [
  {
    "level": "success",
    "text": "Extension deleted."
  }
]
```

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.
to_extensions/{{ID}}/delete

Deprecated since version ATCv4: Use the DELETE method of to_extensions/{{ID}} instead.

**POST**

Deletes a Traffic Ops extension definition. This does **not** delete the actual extension file.

**Auth. Required** Yes

**Roles Required** None

**Response Type** undefined

### Request Structure

Table 279: 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>

# 559: Request Example

```
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

# 560: 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
```

---

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.
traffic_monitor/stats

Deprecated since version ATCv4.

**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

```
application/json response
```

**Request Structure**

No parameters available.
Response Structure

**aaData** An array of data points of some kind

# 561: 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, 03 Dec 2018 14:44:14 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: yRHVMHW+Y78HgaU/UVcrcADq9Jw3ScP+IQEEVqy3R/ 0A75WM22pmGDECDkDp7crWckabMntHRIfaf/6hWJPoQ==
Content-Length: 57

{
  "aaData": [
    [
      "0",
      "ALL",
      "ALL",
      "true",
      "ALL",
      "0",
      "0"
    ]
  ]
  "alerts": [
    {
      "level": "warning",
      "text": "This endpoint and its functionality is deprecated, and will be removed in the future"
    }
  ]
}
```

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 280: 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>

# 562: Request Structure

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

# 563: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly

(continues on next page)
GET

Auth. Required  Yes
Roles Required  None
Response Type  Array

Request Structure

Table 281: 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>

# 564: 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

# 565: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: EH8jo8OrCu79Tz9xpGT3YRyKJ/p2NcTmbS3huwtgRByHz9H6q2LQja59RIPaVSq3ZxsU6QhTaox5nBkQ9LPSAA==
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",      "description": "Location for Traffic Control Component Servers",      "useInTable": "cachegroup"   }  ]}

types/trimmed

Deprecated since version ATCv4: This endpoint and all of its functionality is deprecated. All of the information it can return can be more completely obtained with types.

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.
Warning: This endpoint is of limited use because it doesn’t tell you what the type of each Type is, which describes the types of objects that it can describe. No, I did not just have a stroke while writing this.

Auth. Required Yes

Roles Required None

Response Type Array

Request Structure

No parameters available

Response Structure

name The name of the type

# 566: Response Example

```
HTTP/1.1 200 OK
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Encoding: gzip
Content-Length: 389
Content-Type: application/json
Date: Fri, 31 Jan 2020 18:09:29 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; expires=Fri, 31 Jan 2020 22:09:29 GMT;
    path=/; HttpOnly
Vary: Accept-Encoding

{
    "alerts": [
        {
            "level": "warning",
            "text": "This endpoint is deprecated, please use '/types' instead"
        }
    ],
    "response": [
        {
            "name": "AAAA_RECORD"
        },
        {
            "name": "ANY_MAP"
        },
        {
            "name": "A_RECORD"
        }
    ]
}
```
(continues on next page)
{ "name": "BIND" },
{ "name": "CCR" },
{ "name": "CHECK_EXTENSION_BOOL" },
{ "name": "CHECK_EXTENSION_NUM" },
{ "name": "CHECK_EXTENSION_OPEN_SLOT" },
{ "name": "CLIENT_STEERING" },
{ "name": "CNAME_RECORD" },
{ "name": "CONFIG_EXTENSION" },
{ "name": "DNS" },
{ "name": "DNS_LIVE" },
{ "name": "DNS_LIVE_NATNL" },
{ "name": "EDGE" },
{ "name": "EDGE_LOC" },
{ "name": "ENROLLER" },
{ "name": "GRAFANA" },
{ "name": "HEADER_REGEXP" },
"name": "HOST_REGEXP",
},
{
"name": "HTTP",
},
{
"name": "HTTP_LIVE",
},
{
"name": "HTTP_LIVE_NATNL",
},
{
"name": "HTTP_NO_CACHE",
},
{
"name": "INFLUXDB",
},
{
"name": "MID",
},
{
"name": "MID_LOC",
},
{
"name": "ORG",
},
{
"name": "ORG_LOC",
},
{
"name": "PATH_REGEXP",
},
{
"name": "RASCAL",
},
{
"name": "RESOLVE4",
},
{
"name": "RESOLVE6",
},
{
"name": "RIAK",
},
{
"name": "STATISTIC_EXTENSION",
},
{
"name": "STEERING"


```
{"name": "STEERING_GEO_ORDER"},
{"name": "STEERING_GEO_WEIGHT"},
{"name": "STEERING_ORDER"},
{"name": "STEERING_REGEXP"},
{"name": "STEERING_WEIGHT"},
{"name": "TC_LOC"},
{"name": "TRAFFIC_ANALYTICS"},
{"name": "TRAFFIC_OPS"},
{"name": "TRAFFIC_OPS_DB"},
{"name": "TRAFFIC_PORTAL"},
{"name": "TRAFFIC_STATS"},
{"name": "TR_LOC"},
{"name": "TXT_RECORD"}
]}
```

5.1. Traffic Ops API
Caution: As a username is needed to log in, any administrator or application must necessarily know the current username at any given time. Thus it’s generally better to use the `username` query parameter of a GET request to `users` instead.

Retrieves the details of 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
- `id` An integral, unique identifier for this user
- `lastUpdated` The date and time at which the user was last modified, in an ISO-like 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

username The user's username

# 567: 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: HQwu9FxPyinXSVFK5+wpEhSxU60KbqXuokFbMz3OoerOoM5ZpWpglsHz7mRch8VAw0dzwsJzpPJivj
X-Server-Name: traffic_ops_golang/
Date: Thu, 13 Dec 2018 15:14:45 GMT
Content-Length: 382

{
  "response": {
    "username": "admin",
    "localUser": true,
    "addressLine1": null,
    "addressLine2": null,
    "city": null,
    "company": null,
    "country": null,
    "email": null,
    "fullName": "admin",
    "gid": null,
    "id": 2,
    "newUser": false,
    "phoneNumber": null,
    "postalCode": null,
    "publicSshKey": null,
    "租"
  }
}
```

(continues on next page)
PUT

**Warning:** Assuming the current user’s integral, unique identifier is known, it’s generally better to 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**  Object

Changed in version ATCv4: Starting in ATC version 4, all API versions respond to this endpoint with the updated user information. Prior to that, no response object was returned at all.

**Request Structure**

user  The entire request must be inside a top-level “user” key for legacy reasons  
    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  
    confirmLocalPasswd  An optional ‘confirm’ field in a new user’s password specification. This has no known effect and in fact doesn’t even need to match localPasswd
country The name of the country wherein the user resides

email The user’s email address - cannot be an empty string

Changed in version ATCv4: Prior to version ATCv4, 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”

gid A legacy field only kept for legacy compatibility reasons that used to contain the UNIX group ID of the user - please don’t use this

id The user’s integral, unique, identifier - this cannot be changed

localPasswd Optionally, the user’s password. This should never be given if it will not be changed. An empty string or null can be used to explicitly specify no change.

phoneNumber The user’s phone number

postalCode The user’s postal code

publicSshKey The user’s public encryption key used for the SSH protocol

role The integral, unique identifier of the highest permission Role which will be permitted to the user - this cannot be altered from the user’s current Role

stateOrProvince The state or province in which the user resides

tenantId The integral, unique identifier of the Tenant to which the new user shall belong

uid A legacy field only kept for legacy compatibility reasons that used to contain the UNIX user ID of the user - please don’t use this

username The user’s new username

# 568: Request Example

```
PUT /api/1.4/user/current HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
```

(continues on next page)

---

1 This field cannot be null.
2 This endpoint respects tenancy; a user cannot assign itself to a Tenant that is not the same Tenant to which it was previously assigned or a descendant thereof.
Accept: */*
Cookie: mojolicious=...
Content-Length: 465
Content-Type: application/json

{
  "user": {
    "addressLine1": null,
    "addressLine2": null,
    "city": null,
    "company": null,
    "country": null,
    "email": "admin@infra.trafficops.ciab.test",
    "fullName": "admin",
    "gid": null,
    "id": 2,
    "phoneNumber": null,
    "postalCode": null,
    "publicSshKey": null,
    "role": 1,
    "stateOrProvince": null,
    "tenantId": 1,
    "uid": null,
    "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 legacy field only kept for legacy compatibility reasons that used to contain the UNIX group ID of the user

**id**  An integral, unique identifier for this user

**lastUpdated**  The date and time at which the user was last modified, in an ISO-like format

**newUser**  A meta field with no apparent purpose

**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 legacy field only kept for legacy compatibility reasons that used to contain the UNIX user ID of the user

username  The user’s username

# 569: 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
Date: Thu, 13 Dec 2018 21:05:49 GMT
X-Server-Name: traffic_ops_golang/
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: sHFqZQ4Cv7IIWaIejoAvM2Fr/HSupcX3D16KU/etjw+4jcK9EME3Bq5ohLC+eQ52BDCKW2Ra+AC3TfftworJww==
Content-Length: 478

{ "alerts": [ 
  
  "text": "User profile was successfully updated",
  "level": "success"

  
},

"response": {
  "addressLine1": null,
  "addressLine2": null,
  "city": null,
  "company": null,
  "country": null,

(continues on next page)
"email": "admin@infra.trafficops.ciaab.test",
"fullName": null,
"gid": null,
"id": 2,
"lastUpdated": "2019-10-08 20:14:25+00",
"newUser": false,
"phoneNumber": null,
"postalCode": null,
"publicSshKey": null,
"registrationSent": null,
"role": 1,
"roleName": "admin",
"stateOrProvince": null,
"tenant": "root",
"tenantId": 1,
"uid": null,
"username": "admin"
}

user/current/jobs

Deprecated since version ATCv4: Both request methods supported by this endpoint are im-
plemented (better) by jobs, and in the future that will be the only way to interact with jobs. De-
developers and administrators are encouraged to switch at their earliest convenience.

GET

Retrieves the user’s list of running and pending content invalidation jobs.

Auth. Required Yes

Roles Required None

Response Type Array

Request Structure

Changed in version ATCv4: Prior to version 4 of Traffic Control, the deprecated keyword
query parameter was available to filter jobs in the response. As only one keyword is meaningful,
this was never used and so has been removed.

1 When viewing content invalidation jobs, only those jobs that operate on a Delivery Service visible to the
requesting user’s Tenant will be returned. Likewise, creating a new content invalidation job requires that the target
Delivery Service is modifiable by the requesting user’s Tenant.
# 570: 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 invalidated.
  
  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".

- **deliveryService**: The `xml_id` of the Delivery Service on which this job operates.

- **enteredTime**: The date and time at which the job was created, in the same format as the `last_updated` fields seen throughout other API responses.
  
  Changed in version ATCv4: This used to be in the legacy `YYYY-MM-DD HH:MM:SS` format, but as of Traffic Control version 4 they are standardized to match the format of other date strings in API responses.

- **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 the same format as the `last_updated` fields seen throughout other API responses.
  
  Changed in version ATCv4: This used to be in the legacy `YYYY-MM-DD HH:MM:SS` format, but as of Traffic Control version 4 they are standardized to match the format of other date strings in API responses.

5.1. Traffic Ops API
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

# 571: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: RxFZN2+OvP3HEyp+KlCPDFT74w8PFNjxBjibGIMPhbRjVEb8PhdaF7Gq61w1NRfda4PpTP2t0hei
X-Server-Name: traffic_ops_golang/
Date: Wed, 19 Jun 2019 13:23:18 GMT
Content-Length: 747

{
  "alerts": [
    {
      "text": "This endpoint is deprecated, please use the 'userId' or 'createdBy' query parameters of a GET request to /jobs instead",
      "level": "warning"
    }
  ],
  "response": [{
    "agent": 1,
    "assetType": "file",
    "assetUrl": "http://origin.infra.ciab.test/.*",
    "deliveryService": "demo1",
    "enteredTime": "2019-06-19 13:19:51+00",
    "id": 3,
    "keyword": "PURGE",
    "objectName": null,
    "objectType": null,
    "parameters": "TTL:3h",
    "username": "admin"
  }]
}
```

POST  

Creates a new content revalidation job.
Caution: Creating a content invalidation job immediately triggers a CDN-wide revalidation update. In the case that the global Parameter `use_reval_pending` has a value of exactly "0", this will instead trigger a CDN-wide “Queue Updates”. This means that content invalidation jobs become active immediately at their `startTime` - unlike most other configuration changes they do not wait for a Snapshot or a “Queue Updates”. Furthermore, if the global Parameter `use_reval_pending` is "0", this will cause all pending configuration changes to propagate to all cache servers in the CDN. Take care when using this endpoint.

Auth. Required  Yes
Roles Required  “portal”

Changed in version ATCv3.1.0: For security reasons, the endpoint was reworked so that regardless of tenancy, the “portal” Role or higher is required.

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 it is not necessary to restate the URL but it should be noted that if the origin 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  This can be a string in the legacy YYYY-MM-DD HH:MM:SS format, or a string in RFC 3339 format, or a string representing a date in the same non-standard format as the last_updated fields common in other API responses, or finally it can be a number indicating the number of milliseconds since the Unix Epoch (January 1, 1970 UTC). This date must be in the future, and unlike a POST request to jobs, it must be within two days from the time of creation.
Changed in version ATCv4: Prior to Traffic Control version 4, this used to only accept the legacy YYYY-MM-DD HH:MM:SS date string format, but this constraint has been relaxed. Developers are encouraged to submit date/time strings in either RFC 3339 format or as a numerical Unix timestamp (in milliseconds).

**ttl** Specifies the TTL - in hours - for which the revalidation rule will remain active after `startTime`

**urgent** An optional boolean which, if present and `true`, marks the job as “urgent”, which has no meaning whatsoever, and in fact is not even stored by Traffic Control. So don’t use it.

# 572: Request Example

```
POST /api/1.4/user/current/jobs HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: python-requests/2.20.1
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Cookie: mojolicious=...
Content-Length: 67
Content-Type: application/json

{
    "dsId": 1,
    "startTime": "2019-06-21T00:00:00Z",
    "regex": "/.*",
    "ttl": 3
}
```

Response Structure

Changed in version ATCv4: This method of this endpoint used to only return a successful alert (presuming success), but in ATCv4 a representation of the newly-created content invalidation job was added to the response.

# 573: 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-Encoding: gzip
Content-Type: application/json
Location: https://trafficops.infra.ciab.test/api/1.4/jobs?id=3
```

(continues on next page)
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

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 user’s new username

# 574: 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

# 575: 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=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: sHFqZQ4Cv7IIWaIejoAvM2Fr/HSupcX3D16KU/etjw+4jcK9EME3Bq5ohLC+eQ52BDCtK2Ra+AC3TfFtworJww==
Content-Length: 79

{
  "alerts": [
    {
      "level": "success",
    }
  ]
}
"text": "User profile was successfully updated"
},
{
    "level": "warning",
    "text": "This endpoint is deprecated, please use
'PUT /api/1.4/user/current' instead"
}
]}

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 282: 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>

# 576: 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 Display Name
id  The integral, unique identifier of this Delivery Service

1 Only the Delivery Services visible to the requesting user’s Tenant will appear, regardless of Role or actual ‘assignment’ status.
The `xml_id` which (also) uniquely identifies this Delivery Service

### # 577: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: A1IUM2qkvJkviD0mcEADoCMiy76AWRO/Xnc70ur3CrOlkySXwqxhrhLc3wKlI1926yW+QrTd3nQaVpbX7Rd9wQ==
X-Server-Name: traffic_ops_golang/
Date: Thu, 13 Dec 2018 22:31:44 GMT
Content-Length: 62

{
  "response": [
    {
      "id": 1,
      "displayName": "Demo 1",
      "xmlId": "demo1"
    }
  ]
}
```

**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
# 578: Request Example

```http
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

# 579: 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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512:
X-Server-Name: traffic_ops_golang/
Date: Thu, 13 Dec 2018 15:21:33 GMT
Content-Length: 65

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

user/login/oauth

New in version 1.4.
POST

Authentication of a user by exchanging a code for an encrypted JSON Web Token from an OAuth service. Traffic Ops will POST to the `authCodeTokenUrl` to exchange the code for an encrypted JSON Web Token. It will then decode and validate the token, validate the key set domain, and send back a session cookie.

- **Auth. Required** No
- **Roles Required** None
- **Response Type** undefined

Request Structure

- `authCodeTokenUrl` URL for code-to-token conversion
- `code` Code
- `clientId` Client Id
- `redirectUri` Redirect URI

# 580: Request Example

```
POST /api/1.4/user/login/oauth HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 26
Content-Type: application/json

{
    "authCodeTokenUrl": "https://url-to-convert-code-to-token.example.com",
    "code": "AbCd123",
    "clientId": "oauthClientId",
    "redirectUri": "https://traffic-portal.example.com/sso"
}
```

Response Structure

# 581: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: UdO6T3tMNctnVusDxzRjVwwYOnD7jmnBzPEB9PvOt2bHajTv3SKTPiIZjDzvhU6EX4p-JoG4fA5wH
X-Server-Name: traffic_ops_golang/
Date: Thu, 13 Dec 2018 15:21:33 GMT
Content-Length: 65

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

**user/login/token**

**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** A UUID (Universal Unique Identifier) generated for the user.

**Implementation Detail**

Though not strictly necessary for authentication provided direct database access, the tokens generated for use with this endpoint are compliant with [RFC 4122](http://www.rfc-editor.org/rfc/rfc4122).

# 582: Request Example

```
POST /api/1.3/user/login/token HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
```

(continues on next page)
Response Structure

# 583: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: FuS3TkVosxHtpxRGMJ2on+WNFDyTNSpJxz/Ghl1iT4UCJ2/P0TwUhAGQ3tTx9EfG1A2zg9CNQ1VFpGnYjJZ6NCpg==
X-Server-Name: traffic_ops_golang/
Date: Fri, 20 Sep 2019 15:02:43 GMT
Content-Length: 66

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

user/logout

POST

User logout. Invalidates the session cookie of the currently logged-in user.

Auth. Required  Yes
Roles Required  None
Response Type  undefined

5.1. Traffic Ops API
Request Structure

No parameters available

Response Structure

```
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 21:25:36 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: 6KEdr1ZC512zkO103KwvQE0L7qrJ/+ek6ztymkyY9p8gPFUyYyzGEAJ/Ldb8GY0UBFYmgqe23yWHvTcEsOQMiw==
Content-Length: 61

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

user/reset_password

POST

Sends an email to reset a user’s password.

- **Auth. Required** No
- **Roles Required** None
- **Response Type** undefined

Request Structure

- **email** The email address of the user to initiate password reset
# 585: Request Example

```
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

# 586: 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=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: lKWwVYbgKk1k7b1jnQJzWV5bIjIk+GkooD6SAc3CSexVKvfbL9dgL5iBc/BNNRk2pIU5I/1Gq1dcDLrXsP1ZA==
Content-Length: 109

{
    "alerts": [
        {
            "level": "success",
            "text": "Successfully sent password reset to email 'test@example.com'"
        }
    ]
}
```
GET

Retrieves all requested users.

Auth. Required  Yes

Roles Required  None

Response Type  Array

Request Structure

Table 283: 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 tenant name</td>
</tr>
<tr>
<td>role</td>
<td>no</td>
<td>Return only users belonging to the Role identified by role name</td>
</tr>
<tr>
<td>username</td>
<td>no</td>
<td>Return only the user with this username</td>
</tr>
<tr>
<td>orderBy</td>
<td>no</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>
<tr>
<td>sortOrder</td>
<td>no</td>
<td>Changes the order of sorting. Either ascending (default or “asc”) or descending (“desc”)</td>
</tr>
<tr>
<td>limit</td>
<td>no</td>
<td>The number of results to skip before beginning to return results. Must use in conjunction with limit</td>
</tr>
<tr>
<td>offset</td>
<td>no</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
<tr>
<td>page</td>
<td>no</td>
<td>Return the n\textsuperscript{th} page of results, where “n” is the value of this parameter, pages are limit long and the first page is 1. If offset was defined, this query parameter has no effect. limit must be defined to make use of page.</td>
</tr>
</tbody>
</table>

New in version ATCv4.1: The role query parameter was added in version 4.1 of ATC.

New in version ATCv4.0: The id and username query parameters were added in version 4.0 of ATC.

# 587: 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=...
```

\(^1^\) 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.
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

# 588: 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: *
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: YBJLN8Nb0x0vEclRGlcwCzcIPDhyhLpW56nTJHQ5M5W12WUde2mAKREpaE72nzrfBligELABwJ1sxq
X-Server-Name: traffic_ops_golang/
Date: Thu, 13 Dec 2018 01:03:53 GMT
Content-Length: 391

```json
{
  "response": [
    {
      "username": "admin",
      "registrationSent": null,
      "addressLine1": null,
      "addressLine2": null,
      "city": null,
      "company": null,
      "country": null,
      "email": null,
      "full_name": 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"
    }
  ]
}
```

**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
- **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
# 589: Request Example

```plaintext
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",
   "addressLine1": "22 Mike Wazowski You've Got Your Life Back Lane",
   "city": "Monstropolis",
   "company": "Monsters Inc.",
   "email": "mwazowski@minc.biz",
   "fullName": "Mike Wazowski",
   "localPasswd": "BFFsully",
   "confirmLocalPasswd": "BFFsully",
   "newUser": true,
   "role": 1,
   "tenantId": 1
}
```

**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

---

# 590: 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: Thu, 13 Dec 2018 02:28:27 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019; 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: vDqbaMvg ee0Ids1czqvIWlyDG8WLnCCJdF14Ub05nsE+oJOakkyeZ8odf4d0Zjtqpk01hoVo14H2tj
Content-Length: 520

{
  "alerts": [
    {
      "level": "success",
      "text": "User creation was successful."
    }
  ],
  "response": {
    "registrationSent": null,
    "email": "mwazowski@minc.biz",
    "tenantId": 1,
    "city": "Monstropolis"
  }
}
```

(continues on next page)
users/{{ID}}

GET

Retrieves a specific 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 user to be retrieved</td>
</tr>
</tbody>
</table>

# 591: Request Example

```bash
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

e-mail  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

tenantId  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

# 592: 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: 9vqUmt8fWEuDb+9LQJ4sGbbF4Z0a7uNyBNSWhyzAi3fBUZ5mGhd4Jx5IuS1EqiLZnYeViJ JL8mpR0
X-Server-Name: traffic_ops_golang/
Date: Thu, 13 Dec 2018 17:46:00 GMT
Content-Length: 588

```
{
    "response": [
    {
        "username": "admin",
        "registrationSent": null,
        "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 Full Name",
        "gid": null,
        "id": 2,
        "newUser": false,
        "phoneNumber": "not a real phone number",
        "postalCode": "not a real postal code",
        "publicSshKey": "not a real ssh key",
        "role": 1,
        "rolename": "admin",
        "stateOrProvince": "not a real state or province",
        "tenant": "root",
        "tenantId": 1,
        "uid": null,
        "lastUpdated": "2018-12-13 17:24:23+00"
    }
    ]
}
```

**PUT**

**Auth. Required** Yes

**Roles Required** “admin” or “operations”

**Response Type** Object
Request Structure

Table 285: 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
username The new user's username

# 593: Request Structure

```plaintext
PUT /api/1.1/users/2 HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...
Content-Length: 458
Content-Type: application/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

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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: QKvGSIwSdreMI/OdgWv9WQfI/C1jbxSoQGG0spTGfCVUJ32XNWmhREGzojWsi1W8os8b14TGYeYMUWunf2Ug==
Content-Length: 661

{
"alerts": [

"level": "success",
"text": "User update was successful."
]

(continues on next page)
users/{{ID}}/deliveryservices

**Caution:** This endpoint has several issues related to tenancy and newer Delivery Service fields. For these and other reasons, the assigning of Delivery Services to users is strongly discouraged.

**GET**

Retrieves all Delivery Services assigned to the user.

**Auth. Required** Yes

**Roles Required** None

**Response Type** Array

---

1 While it is totally possible to assign a Delivery Service to a user who’s Tenant does not have permission to own said Delivery Service, users that request this endpoint will only see Delivery Services that their Tenant has permission to see. This means that there’s no real guarantee that the output of this endpoint shows all of the Delivery Services assigned to the user requested, even if the user is requesting their own assigned Delivery Services.
Request Structure

Table 286: 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>

# 595: Request Example

```plaintext
GET /api/1.4/users/2/deliveryservices HTTP/1.1
Host: trafficops.infra.ciab.test
User-Agent: curl/7.47.0
Accept: */*
Cookie: mojolicious=...```

Response Structure

- **active**: A boolean that defines *Active*.
- **anonymousBlockingEnabled**: A boolean that defines *Anonymous Blocking*.
- **cacheurl**: A *Cache URL Expression*
  
  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 *DNS TTL* - 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**: A *Check Path*
- **consistentHashRegex**: A *Consistent Hashing Regular Expression*
  
  New in version 1.4.
- **consistentHashQueryParams**: An array of *Consistent Hashing Query Parameters*
  
  New in version 1.4.

**Caution**: This field will always appear to be null - even when the Delivery Service in question has Consistent Hashing Query Parameters assigned to it.

- **deepCachingType**: The *Deep Caching* setting for this *Delivery Service*
  
  New in version 1.3.
displayName The Display Name

dnsBypassCname A DNS Bypass CNAME

dnsBypassIp A DNS Bypass IP

dnsBypassIp6 A DNS Bypass IPv6

dnsBypassTtl The DNS Bypass TTL

dscp A DSCP to be used within the Delivery Service

escsEnabled A boolean that defines the EDNS0 Client Subnet Enabled setting on this Delivery Service

    New in version 1.4.

edgeHeaderRewrite A set of Edge Header Rewrite Rules

exampleURLs An array of Example URLs

fqPacingRate The Fair-Queuing Pacing Rate Bps

    New in version 1.3.

geoLimit An integer that defines the Geo Limit

geoLimitCountries A string containing a comma-separated list defining the Geo Limit Countries

geoLimitRedirectUrl A Geo Limit Redirect URL

geoProvider The Geolocation Provider

globalMaxMbps The Global Max Mbps

globalMaxTps The Global Max TPS

httpBypassFqdn A HTTP Bypass FQDN

id An integral, unique identifier for this Delivery Service

infoUrl An Info URL

initialDispersion The Initial Dispersion

ipv6RoutingEnabled A boolean that defines the IPv6 Routing Enabled setting on this Delivery Service

lastUpdated The date and time at which this Delivery Service was last updated, in RFC 3339 format

logsEnabled A boolean that defines the Logs Enabled setting on this Delivery Service

longDesc The Long Description of this Delivery Service

longDesc1 The Long Description 2 of this Delivery Service

longDesc2 The Long Description 3 of this Delivery Service

matchList The Delivery Service’s Match List
pattern  A regular expression - the use of this pattern is dependent on the type field (backslashes are escaped)

setNumber  An integer that provides explicit ordering of Match List items - this is used as a priority ranking by Traffic Router, and is not guaranteed to correspond to the ordering of items in the array.

type  The type of match performed using pattern.

maxDnsAnswers  The Max DNS Answers allowed for this Delivery Service

maxOriginConnections  The Max Origin Connections

    New in version 1.4.

midHeaderRewrite  A set of Mid Header Rewrite Rules

missLat  The Geo Miss Default Latitude used by this Delivery Service

missLong  The Geo Miss Default Longitude used by this Delivery Service

multiSiteOrigin  A boolean that defines the use of Use Multi-Site Origin Feature by this Delivery Service

orgServerFqdn  The Origin Server Base URL

originShield  A Origin Shield string

profileDescription  The Description of the Profile with which this Delivery Service is associated

profileId  The ID of the Profile with which this Delivery Service is associated

profileName  The Name of the Profile with which this Delivery Service is associated

protocol  An integral, unique identifier that corresponds to the Protocol used by this Delivery Service

qstringIgnore  An integral, unique identifier that corresponds to the Query String Handling setting on this Delivery Service

rangeRequestHandling  An integral, unique identifier that corresponds to the Range Request Handling setting on this Delivery Service

regexRemap  A Regex Remap Expression

regionalGeoBlocking  A boolean defining the Regional Geoblocking setting on this Delivery Service

remapText  Raw Remap Text

signed  true if and only if signingAlgorithm is not null, false otherwise

signingAlgorithm  Either a Signing Algorithm or null to indicate URL/URI signing is not implemented on this Delivery Service
New in version 1.3.

**sslKeyVersion**  This integer indicates the **SSL Key Version**

**tenantId**  The integral, unique identifier of the **Tenant** who owns this **Delivery Service**

New in version 1.3.

**trRequestHeaders**  If defined, this defines the **Traffic Router Log Request Headers** used by Traffic Router for this **Delivery Service**

New in version 1.3.

**trResponseHeaders**  If defined, this defines the **Traffic Router Additional Response Headers** used by Traffic Router for this **Delivery Service**

New in version 1.3.

**type**  The **Type** of this **Delivery Service**

**typeId**  The integral, unique identifier of the **Type** of this **Delivery Service**

**xmlId**  This **Delivery Service**’s **xml_id**

---

**HTTP/1.1 200 OK**

Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, 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=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-Sha512: YG9PdSw9PAkVLfbTcOfEUbJe14UTkQp2P9x632RbmsbbAQvb1uT5Q1MLJ4OatmEGwWks47NUaRLucqSA==
X-Server-Name: traffic_ops_golang/
Date: Mon, 10 Jun 2019 16:50:25 GMT
Content-Length: 1348

{"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,
}]

(continues on next page)
<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;dnsBypassIp6&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;dnsBypassTtl&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;dscp&quot;</td>
<td>0</td>
</tr>
<tr>
<td>&quot;edgeHeaderRewrite&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;geoLimit&quot;</td>
<td>0</td>
</tr>
<tr>
<td>&quot;geoLimitCountries&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;geoLimitRedirectURL&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;geoProvider&quot;</td>
<td>0</td>
</tr>
<tr>
<td>&quot;globalMaxMbps&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;globalMaxTps&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;httpBypassFqdn&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;id&quot;</td>
<td>1</td>
</tr>
<tr>
<td>&quot;infoUrl&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;initialDispersion&quot;</td>
<td>1</td>
</tr>
<tr>
<td>&quot;ipv6RoutingEnabled&quot;</td>
<td>true</td>
</tr>
<tr>
<td>&quot;lastUpdated&quot;</td>
<td>&quot;2019-06-10 15:14:29+00&quot;</td>
</tr>
<tr>
<td>&quot;logsEnabled&quot;</td>
<td>true</td>
</tr>
<tr>
<td>&quot;longDesc&quot;</td>
<td>&quot;Apachecon North America 2018&quot;</td>
</tr>
<tr>
<td>&quot;longDesc1&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;longDesc2&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;matchList&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;maxDnsAnswers&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;midHeaderRewrite&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;missLat&quot;</td>
<td>42</td>
</tr>
<tr>
<td>&quot;missLong&quot;</td>
<td>-88</td>
</tr>
<tr>
<td>&quot;multiSiteOrigin&quot;</td>
<td>false</td>
</tr>
<tr>
<td>&quot;originShield&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;orgServerFqdn&quot;</td>
<td>&quot;<a href="http://origin.infra.ciab.test">http://origin.infra.ciab.test</a>&quot;</td>
</tr>
<tr>
<td>&quot;profileDescription&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;profileId&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;profileName&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;protocol&quot;</td>
<td>2</td>
</tr>
<tr>
<td>&quot;queryStringIgnore&quot;</td>
<td>0</td>
</tr>
<tr>
<td>&quot;rangeRequestHandling&quot;</td>
<td>0</td>
</tr>
<tr>
<td>&quot;regexRemap&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;regionalGeoBlocking&quot;</td>
<td>false</td>
</tr>
<tr>
<td>&quot;remapText&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;routingName&quot;</td>
<td>&quot;video&quot;</td>
</tr>
<tr>
<td>&quot;signed&quot;</td>
<td>false</td>
</tr>
<tr>
<td>&quot;sslKeyVersion&quot;</td>
<td>1</td>
</tr>
<tr>
<td>&quot;tenantId&quot;</td>
<td>1</td>
</tr>
<tr>
<td>&quot;type&quot;</td>
<td>&quot;HTTP&quot;</td>
</tr>
<tr>
<td>&quot;typeId&quot;</td>
<td>1</td>
</tr>
<tr>
<td>&quot;xmlId&quot;</td>
<td>&quot;demo1&quot;</td>
</tr>
<tr>
<td>&quot;exampleURLs&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;deepCachingType&quot;</td>
<td>&quot;NEVER&quot;</td>
</tr>
<tr>
<td>&quot;fqPacingRate&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;signingAlgorithm&quot;</td>
<td>null</td>
</tr>
<tr>
<td>&quot;tenant&quot;</td>
<td>&quot;root&quot;</td>
</tr>
</tbody>
</table>
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

Changed in version ACTv4.0: Prior to ATC version 4.0, this endpoint allowed for the registration of users with arbitrary Roles. It now restricts the allowed values to identifiers for Roles with at most the same permissions level as the requesting user.

**tenantId**  An optional field containing the integral, unique identifier of the Tenant to which the new user will belong

Changed in version ACTv4.0: Prior to ATC version 4.0, this endpoint allowed for the registration of users with arbitrary Tenants. It now restricts the allowed values to identifiers for Tenants within the requesting user’s Tenant’s permissions.

Changed in version ATCv4.0: As tenancy is no longer truly optional, this field is no longer optional.
# 597: Request Example

```plaintext
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
}
```

# 598: 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: Thu, 13 Dec 2018 22:03:22 GMT
Server: Mojolicious (Perl)
Set-Cookie: mojolicious=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: yvf++0qxv3u0IAYbWLUgJKsx24T60Mi5H9eGTxrKlxnRsHw0PdDIRbTnWTbATB4ak4VU/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 “Request 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.

```markdown
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.

```
{
  "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.

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.
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).

### Example Session

A user makes a request to the /api/1.1/asns endpoint.

```
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.

```
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.

```
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.

```
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
```
Using this cookie, the user can now access their original target - the /api/1.1/asns endpoint...

```
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=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Whole-Content-SHA512: u+Q5X7z/DMTc/VzRGaFlJBa8btA8EC...dnA85HCYTm8vVwsQCv1e+uC1nlA==
X-Server-Name: traffic_ops_golang/

{ "response": {
    "asns": [
        {
            "lastUpdated": "2012-09-17 21:41:22",
            "id": 27,
            "asn": 7015,
        }
    ]
}}
```
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.

### # 600: 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: xRKu2Q7Yj07UA6A6SyxMNNcBpuBcW2/\n\zbzuK05eT2y4V27rXFP/5bSkNPesomJbiO0+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.

# 601: 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=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
Vary: Accept-Encoding
Whole-Content-Sha512: Ff5hO8ZUNUMbwCW0mBuUlsvrSmm/Giijq7O3uLivLZ6V0u6eGom4Jag6Uq1BbbDBnP6AG71lSzdt74TT6NidA==
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)

# 602: 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=...; Path=/; Expires=Mon, 18 Nov 2019 17:40:54 GMT; Max-Age=3600; HttpOnly
```

(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 atstccfg

atstccfg is a tool for generating configuration files server-side on ATC cache servers. It stores its generated/cached files in /tmp/atstccfg_cache/ for re-use.

Warning: atstccfg does not have a stable command-line interface, it can and will change without warning. Scripts should avoid calling it for the time being, as its only intended caller is ORT.

The source code for atstccfg may be found in traffic_ops/ort/atstccfg, and the Go module reference is github.com/apache/trafficcontrol/traffic_ops/ort/atstccfg.

Usage

Options

-a AGE, --cache-file-max-age-seconds AGE
   Sets the maximum age - in seconds - a cached response can be in order to be considered “fresh” - older files will be re-generated and cached. Default: 60

-e ERROR_LOCATION, --log-location-error ERROR_LOCATION
   The file location to which to log errors. Respects the special string constants of github.com/apache/trafficcontrol/lib/go-log. Default: ‘stderr’

-g, --print-generated-files
   If given, the names of files generated (and not proxied to Traffic Ops) will be printed to stdout, then atstccfg will exit.

-i INFO_LOCATION, --log-location-info INFO_LOCATION
   The file location to which to log information messages. Respects the special string constants of github.com/apache/trafficcontrol/lib/go-log. Default: ‘stderr’

-h, --help
   Print usage information and exit.

-l, --list-plugins
   List the loaded plugins and then exit.

-n, --no-cache
   If given, existing cache files will not be used. Cache files will still be created, existing ones just won’t be used.

-P TO_PASSWORD, --traffic-ops-password TO_PASSWORD
   Authenticate using this password - if not given, atstccfg will attempt to use the value of the TO_PASS environment variable.

-r N, --num-retries N
   The number of times to retry getting a file if it fails. Default: 5

-s, --traffic-ops-insecure
   If given, SSL certificate errors will be ignored when communicating with Traffic Ops.

Caution: The use of this option in production environments is discouraged.

-t TIMEOUT, --traffic-ops-timeout-milliseconds TIMEOUT
   Sets the timeout - in milliseconds - for requests made to Traffic Ops. Default: 10000

-u TO_URL, --traffic-ops-url TO_URL
   Request this URL, e.g. https://trafficops.infra.ciab.test/servers/edge/configfiles/ats. If not given, atstccfg will attempt to use the value of the TO_URL environment variable.

-U TO_USER, --traffic-ops-user TO_USER
   Authenticate as the user TO_USER - if not given, atstccfg will attempt to use the value of the TO_USER environment variable.
-v, --version
Print version information and exit.

-w WARNING_LOCATION, --log-location-warning WARNING_LOCATION
The file location to which to log warnings. Respects the special string constants of

Environment Variables

TO_USER
Defines the user as whom to authenticate with Traffic Ops. This is only used if
-U/--traffic-ops-user is not given.

TO_PASS
Defines the password to use when authenticating with Traffic Ops. This is only used if
-P/--traffic-ops-password is not given.

TO_URL
Defines the full URL to be requested. This is only used if -u/--traffic-ops-url
is not given.

6.1.2 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:

- github.com/apache/trafficcontrol/lib/go-tc
- github.com/kelseyhightower/envconfig
- golang.org/x/net/publicsuffix

Theoretically, if you downloaded the Traffic Control repository properly (into $GOPATH/src/github.
com/apache/trafficcontrol), this will already be satisfied.
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.

See also:

Contributions to `genConfigRoutes.py` should follow the ATC Python contribution guidelines.

### Usage

`traffic_ops/testing/compare/compare.go`

```bash
```

**--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 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 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:

```
# 603: 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. 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

```
# 604: Sample Script to Build and Run

sudo docker build . -f traffic_ops/testing/compare/Dockerfile -t 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.

6.1.3 Apache-TrafficControl Package

Contributing

When contributing Python code to Apache Traffic Control, please ensure that it passes the provided Pylint configuration file (traffic_control/clients/python/pylint.rc).
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

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 conve-
nience 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
temporarily 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

# 605: Endpoint with no URL parameters and no query pa-
rameters

```python
@api_request(u'get', u'cdns', (u'1.1', u'1.2',))
def get_cdns(self):
    pass
```

# 606: End-point with URL parameters and no query param-
eters

```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
```

# 607: 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
```

# 608: End-point with URL parameters and query param-
eters

```python
@api_request(u'get', u'deliveryservices/xmlId/{xml_id}/sslkeys',
             (u'1.1', u'1.2',))
```
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)`

`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
assign_deliveryservice_servers_by_names (xml_id=None, data=None)
Assign servers by name to a Delivery Service by xmlId. deliveryservices/{{xml_id}}/servers :param xml_id: The XML Id of the delivery service :type xml_id: str :param data: The required data to assign servers to a delivery service :type data: Dict[str, Any] :rtype: Tuple[Dict[str, Any], requests.Response] :raises: Union[LoginError, OperationError]

assign_federation_resolver_to_federations (federation_id=None, data=None)
Create one or more federation / federation resolver assignments. federations/{{ID}}/federation_resolvers :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] :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response] :raises: Union[LoginError, OperationError]

assign_parameter_to_profile_ids (data=None)
Create one or more parameter / profile assignments. profileparameter :param data: The data to assign :type data: Union[Dict[str, Any], List[Dict[str, Any]]] :rtype: Tuple[Dict[str, Any], requests.Response] :raises: Union[LoginError, OperationError]

assign_profile_to_parameter_ids (data=None)
Create one or more profile / parameter assignments. profileparameter :param data: The data to assign :type data: Union[Dict[str, Any], List[Dict[str, Any]]] :rtype: Tuple[Dict[str, Any], requests.Response] :raises: Union[LoginError, OperationError]

associate_paramater_to_profile (data=None)

associate_parameters_by_profile_id (profile_id=None, data=None)

associate_parameters_by_profile_name (profile_name=None, data=None)
Associate Parameters to a Profile by Name. profiles/name/{{name}}/parameters :param profile_name: The profile name :type profile_name: str :param data: The parameter data to associate :type data: Union[Dict[str, Any], List[Dict[str, Any]]] :rtype: Tuple[Dict[str, Any], requests.Response] :raises: Union[LoginError, OperationError]

base_url
Returns the base url. (read-only)

**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
:program cache_group_id: The Cache Group Id :type cache_group_id: int
:program data: The update action. QueueUpdateRequest() can be used for this argument also.

cdns_queue_update (cdn_id=None, data=None)
Queue Updates by CDN Id. cdns/{{ID}}/queue_update
:program cdn_id: The CDN Id :type cdn_id: int
:program data: The update action. QueueUpdateRequest() can be used for this argument also.

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}}
:program new_profile_name: The name of profile to copy to :type new_profile_name: String
:program copy_profile_name: The name of profile copy from :type copy_profile_name: String
:program data: The update action. QueueUpdateRequest() can be used for this argument also.

create_asn (data=None)
Create ASN asns :program data: The parameter data to use for cachegroup creation.

create_cache_group_fallbacks (data=None)
Creates fallback configuration for the cache group. New fallbacks can be added only via POST.
cache fallbacks :program data: The update action. QueueUpdateRequest() can be used for this argument also.

create_cachegroups (data=None)
Create a Cache Group cachegroups :program data: The parameter data to use for cachegroup creation.

create_capability (data=None)
Creates a capability capabilities :program data: See API page for more information on accepted request body data
create_cdn (data=None)

create_cdn_dns_sec_keys (data=None)

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. :param federation_id: Federation ID :type federation_id: int :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_invalidation_job (data=None)
Invalidating 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. :param user/current/jobs: 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_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[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[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.

**create_server** *(data=None)*

Create a new Server.

- **servers** :param `data`: The parameter data to use for server creation

**create_servercheck** *(data=None)*

Post a server check result to the serverchecks table.

- **servercheck** :param `data`: The parameter data to use for server creation

**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.

**create_tenant** *(data=None)*

Create a tenant.

- **tenants** :param `data`: The update action. QueueUpdateRequest() can be used for this argument also.

**create_to_extension** *(data=None)*

Creates a Traffic Ops extension.

- **to_extensions** :param `data`: The update action. QueueUpdateRequest() can be used for this argument also.

**create_user** *(data=None)*

Create a user.

- **users** :param `data`: The update action. QueueUpdateRequest() can be used for this argument also.

**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.

**delete_asn** *(asn_id=None)*

Delete ASN.

- **to-api-asns-id** :param `asn_id`: The ID of the ASN to delete
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]

Raises  
Union[LoginError, OperationError]

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]

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]

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]

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]

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]

delete_delivery_service_user_link(delivery_service_id=None, user_id=None)

Removes 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]

delete_deliveryservice_by_id(delivery_service_id=None)

Allows user to delete a delivery service.  
deliveryservices/{{ID}}  
:param deli-

**delete_deliveryservice_regex_by_regex_id**


**delete_deliveryservice_servers_by_id**


**delete_deliveryservice_ssl_keys_by_xml_id**


**delete_division**


**delete_federation_in_cdn**


**delete_federation_resolver**


**delete_federation_user**

Delete one or more federation / user assignments. federations/{{ID}}/users/{{userID}} :param federation_id: Federation ID :type fed-
delete_origins(query_params=None)
Updates origins associated with a delivery service origins :param query_params:

delete_parameter(parameter_id=None)

delete_physical_location(physical_location_id=None, query_params=None)

delete_profile_by_id(profile_id=None)

delete_profile_parameter_association_by_id(profile_id=None, parameter_id=None)

delete_region(query_params=None)

delete_server_by_id(server_id=None)

delete_staticdnsentries(query_params=None)
**delete_to_extension** *(extension_id=None)*
Deletes a Traffic Ops extension. 

```python
```

**generate_deliveryservice_ssl_keys** *(data=None)*
Generate an SSL certificate. 

```python
```

**generate_deliveryservice_url_signature_keys** *(xml_id=None)*
Generate URL Signature Keys for a Delivery Service by xmlId. 

```python
```

**generate_iso** *(data=None)*
Generate an ISO 

```python
```

**get_all_cachegroup_parameters** *
A collection of all cache group parameters. 

```python
```

**get_all_deliveryservice_servers** *(*args, **kwargs)*
Get all servers attached to all delivery services via the Traffic Ops API. 

```python
```

**get_api_capabilities** *(query_params=None)*
Get all API-capability mappings 

```python
```

**get_api_capabilities_by_id** *(id=None)*
Get an API-capability mapping by ID 

```python
```

**get_asn_by_id** *(asn_id=None)*
Get ASN by ID 

```python
```

**get_asns** *(query_params=None)*
Get ASNs. 

```python
```

**get_associated_profiles_by_parameter_id** *(parameter_id=None)*
Get all Profiles associated to a Parameter by Id. 

```python
parameters/{{ID}}/profiles
```

:param parameter_id: The parameter id
:rtype: Tuple[Dict[str, Any], requests.Response]
:raises: Union[LoginError, OperationError]

get_authenticated_user()  
Retrieves the profile for the authenticated user. 

```python
user/current
```

:rtype: Tuple[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]

get_authenticated_user_jobs()  
Retrieves the user’s list of jobs. 

```python
user/current/jobs
```

:rtype: Tuple[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]

get_cache_group_fallbacks(query_params=None)

Retrieve fallback related configurations for a cache group

```python
cachegroup_fallbacks
```

Parameters

```python
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)

Retrieves statistics about the CDN.

```python
cache_stats
```

:param query_params: See API page for more information on accepted params
:rtype: Tuple[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]

get_cachegroup_by_id(cache_group_id=None)

Get a Cache Group by Id.

```python
cachegroups/{{ID}}
```

:param cache_group_id: The cache group Id
:rtype: Tuple[Dict[str, Any], requests.Response]
:raises: Union[LoginError, OperationError]

get_cachegroup_parameters(cache_group_id=None)

Get a cache groups parameters

```python
cachegroups/{{ID}}/parameters
```

:param cache_group_id: The cache group Id
:rtype: Tuple[Dict[str, Any], requests.Response]
:raises: Union[LoginError, OperationError]

get_cachegroup_parameters_by_id(parameter_id=None)

Get a cache groups parameter by its Id

```python
cachegroup/{{parameter_id}}/parameter
```

:param parameter_id: The parameter Id
:rtype: Tuple[Dict[str, Any], requests.Response]
:raises: Union[LoginError, OperationError]

get_cachegroup_unassigned_parameters(cache_group_id=None)

Get a cache groups unassigned parameters

```python
cachegroups/{{id}}/unassigned_parameters
```

:param cache_group_id: The cache group Id
:rtype: Tuple[Dict[str, Any], requests.Response]
:raises: Union[LoginError, OperationError]

get_cachegroups(query_params=None)

Get Cache Groups.

```python
cachegroups
```

:rtype: Tuple[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]
get_capabilities(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 dnsseckeys info for :type cdn_name: String :rtype: Tuple[Dict[str, Any], requests.Response] :raises: Union[LoginError, OperationError]

get_cdn_health_by_name(cdn_name=None)
Retrieves the health of all locations (cache groups) for a given CDN cdns/{{name}}/health :param cdn_name: The CDN name to find health for :type cdn_name: String :rtype: Tuple[Dict[str, Any], requests.Response] :raises: Union[LoginError, OperationError]

get_cdn_monitoring_info(cdn_name=None)

get_cdn_routing_info(cdn_name=None)

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 cdns/{{cdn}}/configfiles/ats/{{filename}} :param cdn_name: The cdn name to get config files for :type cdn_name: String :param config_file: The config file name to retrieve for host :type config_file: String :rtype: Tuple[Dict[str, Any], requests.Response] :raises: Union[LoginError, OperationError]

get_cdn_ssl_keys(cdn_name=None)
Returns ssl certificates for all Delivery Services that are a part of the CDN. cdns/name/{{name}}/sslkeys :param cdn_name: The CDN name to find ssl keys for
get_cdns()  

get_cdns_capacity()  
Retrieves the aggregate capacity percentages of all locations (cache groups) for a given CDN. **cdns/capacity** :rtype: Tuple[Dict[str, Any], requests.Response] :raises: Union[LoginError, OperationError]

get_cdns_domains()  
Retrieves the different CDN domains **cdns/domains** :rtype: Tuple[Dict[str, Any], requests.Response] :raises: Union[LoginError, OperationError]

get_cdns_health()  
Retrieves the health of all locations (cache groups) for all CDNs **cdns/health** :rtype: Tuple[Dict[str, Any], requests.Response] :raises: Union[LoginError, OperationError]

get_cdns_routing()  
Retrieves the aggregate routing percentages of all locations (cache groups) for a given CDN. **cdns/routing** :rtype: Tuple[Dict[str, Any], requests.Response] :raises: Union[LoginError, OperationError]

get_cdns_usage()  

get_change_logs()  
Retrieve all change logs from traffic ops **logs** :rtype: Tuple[Dict[str, Any], requests.Response] :raises: Union[LoginError, OperationError]

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. **cdns/[[name]]/snapshot** :param

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. deliveryservices/{{ID}}/capacity :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_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. deliveryservices/{{ID}}/state :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_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. deliveryservices/{{ID}}/health :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_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 RegExes for a Delivery Service by Id. deliveryservices/{{ID}}/regexes :param delivery_service_id: The delivery service Id :type delivery_service_id:
get_deliveryservice_regexes_by_regex_id(delivery_service_id=None, regex_id=None)
Retrieves a regex for a specific delivery service.

deliveryservices/{{ID}}/regexes/{{rID}}

: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
:rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]

get_deliveryservice_servers(delivery_service_id=None)
Retrieves properties of CDN EDGE or ORG servers assigned to a delivery service.

deliveryservices/{{ID}}/servers

:param delivery_service_id: The delivery service Id
:type delivery_service_id: int
:rtype: Tuple[Dict[str, Any], requests.Response]
:raises: Union[LoginError, OperationError]

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
:type xml_id: str
:param query_params: The url query parameters for the call
:type query_params: Dict[str, Any]
:rtype: Tuple[Dict[str, Any], requests.Response]
:raises: Union[LoginError, OperationError]

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

:param delivery_service_id: The delivery service Id
:type delivery_service_id: int
:rtype: Tuple[Dict[str, Any], requests.Response]
:raises: Union[LoginError, OperationError]

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 RegExes for all Delivery Services.

deliveryservices_regexes

:rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]

get_deliveryserviceserver(query_params=None)
Retrieves delivery service / server assignments. (Allows pagination and limits)

deliveryserviceserver

:param query_params: The required 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_division_by_id(division_id=None)
Get a division by division id

divisions/{{ID}}

:param division_id: The division id to retrieve
:type division_id: int
:rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]
get_divisions()  
Get all divisions.  
\[
\]

get_federation_delivery_services(federation_id=None)  
Retrieves delivery services assigned to a federation.  
\[
\]

get_federation_for_cdn_by_id(cdn_name=None, federation_id=None)  
Retrieves a federation for a cdn.  
\[
\]

get_federation_resolvers(query_params=None)  
Get federation resolvers.  
\[
\]

get_federation_resolvers_by_id(federation_id=None)  
Retrieves federation resolvers assigned to a federation.  
\[
\]

get_federation_users(federation_id=None)  
Retrieves users assigned to a federation.  
\[
\]

get_federations()  
Retrieves a list of federation mappings (aka federation resolvers) for a the current user.  
\[
\]

get_federations_for_cdn(cdn_name=None)  
Retrieves a list of federations for a cdn.  
\[
\]

get_hwinfo()  
Get hwinfo for servers.  
\[
\]

get_job_by_id(job_id=None)  
Get a job by ID (currently limited to invalidate content (PURGE) jobs).  
\[
\text{jobs/{{ID}}}
\]
get_jobs

Get all jobs (currently limited to invalidate content (PURGE) jobs) sorted by start
time (descending).

:param query_params=None :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]],

get_origins

Get origins associated with the delivery service

:param query_params: The optional url query parameters for
the call :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]],

get_osversions

Get all OS versions for ISO generation and the directory where the kickstarter files
are found. The values are retrieved from osversions.json found in either
/var/www/files or in the location defined by the kickstart.files.location parameter
(if defined).

:get_osversions() :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]],

get_parameters

Get all Profile Parameters.

:get_parameters() :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]],

get_parameters_by_profile_id

Get all Parameters associated with a Profile by Id.

:get_parameters_by_profile_id() :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]],

get_parameters_by_profile_name

Get all Parameters associated with a Profile by Name.

:get_parameters_by_profile_name() :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]],

get_pending_snapshot_crconfig

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.

get_physical_location_by_id(physical_location_id=None)
Get Physical Location by id phys_locations/{{ID}}
:param physical_location_id: The id to retrieve
:type physical_location_id: int
:rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]

get_physical_locations(query_params=None)
Get Physical Locations. phys_locations
:type physical_locations: 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. profiles/{{ID}}
:param profile_id: The profile Id
:type profile_id: int
:rtype: 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 profiles/[[profile]]/configfiles/ats/[[filename]]
:param profile_name: The profile name to get config files for
:type profile_name: String
:param config_file: The config file name to retrieve for host
:type config_file: String
:rtype: Tuple[Dict[str, Any], requests.Response]
:raises: Union[LoginError, OperationError]

get_profiles(query_params=None)
Get Profiles. profiles
:type profiles: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]

get_region_by_id(region_id=None)
Get Region by ID regions/{{ID}}
:param region_id: The region id of the region to retrieve
:type region_id: int
:rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]

get_regions(query_params=None)
Get Regions. regions
: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_roles()
Get Roles. roles
:type roles: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]

get_server_by_id(server_id=None)
Get Server by Server ID servers/{{ID}}
:param server_id: The server id to retrieve
:type server_id: int
:rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]

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
:type host_name: String
get_server_delivery_services (server_id=None)
Retrieves all delivery services assigned to the server.
:servers/{{ID}}/deliveryservices
:rtype: Tuple[Dict[str, Any], requests.Response]
:param server_id: The server id to retrieve.
:type server_id: int
:raises: Union[LoginError, OperationError]

get_server_details (name=None)
Get server details from trafficOps.
servers/hostname/{{name}}/details
:param hostname: Server hostname.
:type hostname: String
:rtype: Tuple[Dict[str, Any], List[Dict[str, Any]]]
:raises: Union[LoginError, OperationError]

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.
:type host_name: String
:param config_file: The config file name to retrieve for host.
:type config_file: String
: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
:rtype: Tuple[Dict[str, Any], List[Dict[str, Any]]]
:raises: Union[LoginError, OperationError]

get_server_type_count ()
Retrieves a count of CDN servers by type.
servers/totals
:rtype: Tuple[Dict[str, Any], List[Dict[str, Any]]]
:raises: Union[LoginError, OperationError]

get_server_update_status (server_name=None)
Gets the current update status of a server named.
servers/{{hostname}}/update_status
:param server_name: The (short) hostname of the server for which the update status will be fetched.
:type server_name: String
:rtype: Tuple[Dict[str, Any], requests.Response]
:raises: Union[LoginError, OperationError]

get_servers (query_params=None)
Get Servers.
servers
:param query_params: The optional url query parameters for the call.
:type query_params: Dict[str, Any]
:rtype: Tuple[Dict[str, Any], List[Dict[str, Any]]]
:raises: Union[LoginError, OperationError]

get_static_dns_entries ()
Get Static DNS Entries.
staticdnsentries
:rtype: Tuple[Dict[str, Any], List[Dict[str, Any]]]
:raises: Union[LoginError, OperationError]

get_staticdnsentries (query_params=None)
Get static DNS entries associated with the delivery service.
staticdnsentries
:param query_params: The optional url query parameters for the call.
:type query_params:
get_statuses()

get_statuses_by_id(status_id=None)

get_system_info()

get_tenant_by_id(tenant_id=None)

get_tenants()

get_to_extensions()

get_traffic_monitor_cache_stats()

get_trimmed_physical_locations()

get_trimmed_profiles()

get_type_by_id(type_id=None)
Get Data Type with the given type id types/{{ID}} :param type_id: The ID of the type to retrieve :type type_id: int :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response] :raises: Union[LoginError, OperationError]

get_types(query_params=None)
Get Data Types.  

```py
get_types_only_names()
```

Get Data Types with only the Names  

```py
get_unassigned_profiles_by_parameter_id(parameter_id=None)
```

Retrieves all profiles NOT assigned to the parameter.  

```py
get_unassigned_parameters_by_profile_id(profile_id=None)
```

Get all Parameters associated with a Profile by Id.  

```py
get_user_by_id(user_id=None)
```

Retrieves user by ID.  

```py
get_users()
```

Retrieves all users.

logged_in

Read-only property of to determine if user is logged in to Traffic Ops.  

```py
login(username, password)
```

Login to the Traffic Ops API.  

```py
replace_authenticated_user(data=None)
```

Updates the currently authenticated user.  

```py
servers_queue_update(server_id=None, data=None)
```

Queue Updates by Server Id.
dateRequest() can be used for this argument also. :type data: Dict[str, Any] :type: Tuple[Dict[str, Any], requests.Response] :raises: Union[LoginError, OperationError]

snapshot_crconfig(cdn_name=None)

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_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] :type: Tuple[Dict[str, Any], requests.Response] :raises: Union[LoginError, OperationError]

update_cachegroups(cache_group_id=None, data=None)

update_cdn_by_id(cdn_id=None, data=None)

update_coordinates(query_params=None, data=None)

update_deliveryservice_by_id(delivery_service_id=None, data=None)
Update a Delivery Service by Id. deliveryservices/{{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]
update_deliveryservice_regexes(delivery_service_id=None, regex_id=None, query_params=None)

Update a regex for a delivery service

deliveryservices/{{ID}}/regexes/{{rID}}

: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 a Delivery Service.

deliveryservices/{{ID}}/safe

: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[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]
:raises: Union[LoginError, OperationError]

update_division(division_id=None, query_params=None)

Update a division by division id

divisions/{{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[Union[Dict[str, Any], List[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.

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

:rtype: 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

:rtype: 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

parameters/{{ID}}

:param parameter_id: The parameter id to update
:type parameter_id: int

:rtype: 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

phys_locations/{{ID}}

:param physical_location_id: The physical location id to update
:type physical_location_id: int

:rtype: Tuple[Dict[str, Any], requests.Response]
:raises: Union[LoginError, OperationError]
update_profile_by_id

```
update_profile_by_id(profile_id=None, data=None)
```

Update Profile by Id. **profiles/[[ID]]**

- **profile_id**: The profile Id
- **data**: The parameter data to edit

> :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]

```
:raises: Union[LoginError, OperationError]
```

update_region

```
update_region(region_id=None)
```

Update a region **regions/[[ID]]**

- **region_id**: The region to update

> :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]

```
:raises: Union[LoginError, OperationError]
```

update_server_by_id

```
update_server_by_id(server_id=None, data=None)
```

Update a Server by Id. **servers/[[ID]]**

- **server_id**: The server Id
- **data**: The parameter data to edit

> :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]

```
:raises: Union[LoginError, OperationError]
```

update_server_status_by_id

```
update_server_status_by_id(server_id=None, data=None)
```

Update server_status by Id. **servers/[[ID]]/status**

- **server_id**: The server Id

> :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]

```
:raises: Union[LoginError, OperationError]
```

update_staticdnsentries

```
update_staticdnsentries(data=None, query_params=None)
```

Update static DNS entries associated with the delivery service **staticdnsentries**

- **data**: The update action. QueueUpdateRequest() can be used for this argument also.
- **query_params**: The optional url query parameters for the call

> :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]

```
:raises: Union[LoginError, OperationError]
```

update_tenant

```
update_tenant(tenant_id=None)
```

Update a tenant **tenants/[[ID]]**

- **tenant_id**: The tenant to update

> :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]

```
:raises: Union[LoginError, OperationError]
```

update_user_by_id

```
update_user_by_id(user_id=None, data=None)
```

Update a user. **users**

- **data**: The user update data payload

> :rtype: Tuple[Union[Dict[str, Any], List[Dict[str, Any]]], requests.Response]

```
:raises: Union[LoginError, OperationError]
```

validate_parameter_exists

```
validate_parameter_exists(data=None)
```

Validate that a Parameter exists. **parameters/validate**

- **data**: The parameter data to use for parameter validation

> :rtype: Tuple[Dict[str, Any], requests.Response]

```
:raises: Union[LoginError, OperationError]
```

exception trafficops.InvalidJSONError(*args, resp=None)

Bases: ValueError

validating parameter exists...
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 specification as per `str.format()` e.g. `cachegroups/{id}` or `cachegroups/{id:d}`

- **supported_versions** *(Tuple[str]*) – A tuple of API versions that this route supports

**Returns**

```
rtype int: A new function that replaces the original function with a boilerplate execution process.
```

**Return type**

```
Callable[str, Dict[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)
```
This class represents a login session with a generic REST API server. It provides base functionality inherited by `TOSession`. 

Chapter 6. Tools
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 str

api_version
Returns the api version. (read-only)
Returns The api version from which this instance will request endpoints.
Return type str

close()
Close and cleanup the requests Session object.
Returns None
Return type NoneType

create()
Create the requests.Session to communicate with the RESTful API.
Returns None
Return type NoneType

delete(api_path, *args, **kwargs)
Perform HTTP DELETE 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]

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 `str.format()` e.g. `cachegroups/{id}` or `cachegroups/{id:d}` :type api_path: str

:param kwargs: 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 :type kwargs: Dict[str, Any]


: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 `str.format()` e.g. `cachegroups/{id}` or `cachegroups/{id:d}` :type api_path: str

:param kwargs: 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:

```python
def query_params(kwargs):
    return: Python data structure distilled from JSON from the API request.
    :raises: Union[LoginError, OperationError]
```

### patch (api_path, *args, **kwargs)
Perform HTTP PATCH requests

- **api_path** (str) – The path to the API endpoint 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:

```python
def query_params(kwargs):
    return: Python data structure distilled from JSON from the API request.
    :raises: Union[LoginError, OperationError]
```

### post (api_path, *args, **kwargs)
Perform HTTP POST requests

- **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:

```python
def query_params(kwargs):
    return: Python data structure distilled from JSON from the API request.
    :raises: Union[LoginError, OperationError]
```

### put (api_path, *args, **kwargs)
Perform HTTP PUT requests

- **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}

Python data structure distilled from JSON from the API request.


- **Raises** Union[LoginError, OperationError]
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]

**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 separator
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.

### 6.1.4 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 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 optionally start with `http://` or `https://` (case insensitive), but typically this is unnecessary. 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 otherwise 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. Overridden 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 `-p/--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 `-p/--pretty`.
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**

```python
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='\t')
    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)

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
```
to_access.patch()
Entry point for topatch
Returns The program’s exit code

to_access.post()
Entry point for topost
Returns The program’s exit code

to_access.put()
Entry point for toput
Returns The program’s exit code

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

6.1.5 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  
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  
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  
An optional flag which, if given, sets the port to which traffic_vault_util will try to connect to Riak. Default: 8087

--vault_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.

---

1 These problems are all tracked by GitHub Issue #3261.
CHAPTER 7

FAQ

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 Group**

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 geographic 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 usually 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 (or Deep Coverage Zone File, when applicable).

See also:

For a more complete description of Cache Groups, see the Cache Groups overview section.

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.

ccontent 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.

# 611: Example Coverage Zone File

```json
{
  "coverageZones": {
    "cache-group-01": {
      "network6": [
        "1234:5678::/64",
        "1234:5679::/64"
      ],
    }
  }
}
```

(continues on next page)
"network": [
    "192.168.8.0/24",
    "192.168.9.0/24"
  ]
}

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.

# 612: Example Deep Coverage Zone File

```
{ "coverageZones": {
  "cache-group-01": {
    "network6": [
      "1234:5678::/64",
      "1234:5679::/64"
    ],
    "network": [
      "192.168.8.0/24",
      "192.168.9.0/24"
    ],
    "caches": [
      "edge"
    ]
  }
}
```

Delivery Service

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.

Delivery Service required capabilities Delivery Service required capabilities are capabilities, which correlate to server capabilities, that are required in order to assign a server to a delivery service.

Division

Divisions 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]], federations/[[ID]]/deliveryservices,
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 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.

   # 613: Client Requests Content from its Forward Proxy

   ```plaintext
   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.

      # 614: The Forward Proxy Requests Content from the Origin Server

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

   2. The origin server responds with the requested content.

      # 615: The Origin Server’s Response

      ```plaintext
      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
      ```

      (continues on next page)
3. The proxy sends this on to the client, optionally adding a Via: header to indicate that the request was serviced by proxy.

# 616: 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
Content-Type: text/html; charset=UTF-8
Age: 0
Via: http/1.1 cache01.cdn.kabletown.net (ApacheTrafficServer/4.2.1 [uScSsSfUpSeN:t cCSi p SS])
Server: ATS/4.2.1

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

If, however, the requested content was in the cache the proxy responds to the client with the previously retrieved result

# 617: 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

<!DOCTYPE html><html><body>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 serve 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 also:**

See *Configuring Traffic Server* for a Python implementation of ORT that is (theoretically) compatible with the one actually provided in Apache Traffic Control releases.

**Parameter**

**Parameters** Typically refers to a line in a configuration file, but in practice can represent any arbitrary configuration option.

**See also:**

The *Profiles and Parameters* overview section.

**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**

**Physical Locations** 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**
Profiles  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/create_tables.sql.

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.

See also:
The Profiles and Parameters overview section.

Queue
Queue Updates
Queue Server Updates  Queuing Updates is an action that signals to various ATC components - most notably cache servers - that any configuration changes that are pending are to be applied now. Specifically, Traffic Monitor and Traffic Router are updated through a CDN Snapshot, and not Queued Updates. In particular, ORT will notice that the server on which it’s running has new configuration, and will request the new configuration from Traffic Ops.

Updates may be queued on a server-by-server basis (in Traffic Portal’s Servers view), a Cache Group-wide basis (in Traffic Portal’s Cache Groups view), or on a CDN-wide basis (in Traffic Portal’s CDNs view). Usually using the CDN-wide version is easiest, and unless there are special circumstances, and/or the user really knows what he or she is doing, it is recommended that the full CDN-wide Queue Updates be used.

This is similar to taking a CDN Snapshot, but this configuration change affects only servers, and not routing.

That seems like a vague difference because it is - in general the rule to follow is that changes to Profiles and Parameters requires only updates be queued, changes to the assignments of cache servers to Delivery Services requires both a Snapshot and a Queue Updates, and changes to only a Delivery Service itself (usually) entails a Snapshot only. These aren’t exhaustive rules, and a grasp of what changes require which action(s) will take time to form. In general, when doing both Queuing Updates as well as taking a CDN Snapshot, it is advisable to first Queue Updates and then take the Snapshot, as otherwise Traffic Router may route clients to Edge-tier cache servers that are not equipped to service their request(s). However, when modifying the assignment(s) of cache servers to one or more Delivery Services, a Snapshot ought to be taken before updates are queued.

Warning:  Updates to Parameters with certain Config File values may require running ORT in a different mode, occasionally manually. Though the server may appear to no longer have pending updates in these cases, until this manual intervention is
Region
Regions  A group of Physical Locations.
reverse proxy
reverse proxies  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 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.
# 618: Client Requests Content from the *Reverse Proxy*

<table>
<thead>
<tr>
<th>GET /foo/bar/fun.html HTTP/1.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host: www-origin-cache.cdn.com</td>
</tr>
</tbody>
</table>

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*

```
# 619: *Reverse Proxy* Requests Content from the *Origin Server*

<table>
<thead>
<tr>
<th>GET /foo/bar/fun.html HTTP/1.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host: <a href="http://www.origin.com">www.origin.com</a></td>
</tr>
</tbody>
</table>
```

2. The *origin server* responds with the requested content

```
# 620: *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: "1a008f-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.

```
# 621: 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: "1a008f-2d-50a3559482cc0"
```
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.

### # 622: The Reverse Proxy Provides a 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: 39711
Via: http/1.1 cache01.cdn.kabletown.net (ApacheTrafficServer/4.2.1)
      [uScSsSfUpSeN:t cCSi p sS])
Server: ATS/4.2.1

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

## Role

**Roles** Permissions *Roles* define the operations a user is allowed to perform, and are currently an ordered list of permission levels.

## Server Capability

**Server Capabilities** A *Server Capability* (not to be confused with a “Capability”) expresses the capacity of a *cache server* to serve a particular kind of traffic. For example, a *Server Capability* could be created named “RAM” to be assigned to *cache servers* that have RAM-disks allocated for content caching. *Server Capabilities* can also be required by *Delivery Services*, which will prevent *cache servers* without that *Server Capability* from being assigned to them. It also prevents *Mid-tier Cache Servers* without said *Server Capability* from being selected to serve upstream requests from those *Edge-tier Cache Servers* assigned to the requiring *Delivery Services*.

## Snapshot

---

8.1. **Glossary**
Snapshots

CDN Snapshot

CDN Snapshots  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

Statuses  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

Tenancy

Tenancies  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.
Python Module Index

t
to_access, 927
traffic_ops_ort, 245
traffic_ops_ort.configuration, 251
traffic_ops_ort.main_routines, 253
traffic_ops_ort.packaging, 255
traffic_ops_ort.to_api, 256
traffic_ops_ort.utils, 259
trafficops, 897
trafficops.__version__, 927
trafficops.utils, 926
Symbols

- api-routes
  traffic_ops command line option, 84
- cfg CONFIG
  admin command line option, 233
- cfg CONFIG_PATH
  traffic_ops command line option, 84
- database database_name
  sync_ts_databases command line option, 163
- days N
  sync_ts_databases command line option, 163
- dbcfg DB_CONFIG_PATH
  traffic_ops command line option, 84
- dir directory
  command line option, 184
- dispersion DISP
  traffic_ops_ort command line option, 246
- dry_run
  traffic_vault_util command line option, 931
- env ENVIRONMENT
  admin command line option, 230
- fixtures FIXTURES
  admin command line option, 233
- help
  create_ts_databases command line option, 162
  sync_ts_databases command line option, 163
- http port
  command line option, 184
- login Dispersion DISP
  traffic_ops_ort command line option, 246
- password password
  create_ts_databases command line option, 162
- plugins
  traffic_ops command line option, 84
- refPasswd PASSWD
  genConfigRoutes.py command line option, 893
- refURL URL
  genConfigRoutes.py command line option, 893
- refUser USER
  genConfigRoutes.py command line option, 893
- ref_passwd PASSWD
  compare command line option, 892
- ref_url URL
  compare command line option, 892
- ref_user USER
  compare command line option, 892
- replication N
  create_ts_databases command line option, 162
- request-headers
-toaccess command line option, 928
-request-payload toaccess command line option, 928
-response-headers toaccess command line option, 928
-retries RETRIES traffic_ops_ort command line option, 246
-rev_prox_disable traffic_ops_ort command line option, 246
-riakcfg TRAFFIC_VAULT_CONFIG_PATH toaccess command line option, 85
-source-password password sync_ts_databases command line option, 163
-source-url URL sync_ts_databases command line option, 163
-source-user username sync_ts_databases command line option, 163
-started filename command line option, 184
-target-password password sync_ts_databases command line option, 163
-target-url URL sync_ts_databases command line option, 163
-target-user username sync_ts_databases command line option, 163
-testPasswd PASSWD genConfigRoutes.py command line option, 894
-test_url URL genConfigRoutes.py command line option, 893
-test_user USER genConfigRoutes.py command line option, 893
-test_passwd PASSWD compare command line option, 892
-test_url URL compare command line option, 892
-test_user USER compare command line option, 892
-to-password PASSWORD toaccess command line option, 928
-to-url URL toaccess command line option, 928
-to-user USERNAME toaccess command line option, 928
-ts_root PATH traffic_ops_ort command line option, 246
-url URL create_ts_databases command line option, 162
-user username create_ts_databases command line option, 162
-vault_action ACTION traffic_vault_util command line option, 931
-vault_ip IP traffic_vault_util command line option, 931
-vault_password PASSWD traffic_vault_util command line option, 932
-vault_port PORT traffic_vault_util command line option, 932
-vault_user USER traffic_vault_util command line option, 932
-version traffic_ops command line option, 85
-wait_for_parents INT traffic_ops_ort command line option, 246
-C, -no-server-configs genConfigRoutes.py command
-P TO_PASSWORD,  
  -traffic-ops-password
TO_PASSWORD
  command line option, 894
-P TO_PASSWORD
  traffic_ops_ort command line option, 246
-U TO_USER, -traffic-ops-user
TO_USER
  command line option, 890
-U TO_USER
  traffic_ops_ort command line option, 246
-V, -version
  compare command line option, 893
-V
  genConfigRoutes.py command line option, 894
-a AGE, -cache-file-max-age-seconds
AGE
  command line option, 890
-a API_VERSION, -api-version
API_VERSION
toaccess command line option, 928
-a API_VERSION
  genConfigRoutes.py command line option, 894
-e ERROR_LOCATION,  
  -log-location-error
ERROR_LOCATION
  command line option, 890
-e ERROR_LOCATION
  traffic_ops_ort command line option, 248
-f FILE, -file FILE
  compare command line option, 892
-f FILE
  toaccess command line option, 928
-f, -full
  toaccess command line option, 928
-g, -print-generated-files
  command line option, 890
-g, -print-generated-files
  toaccess command line option, 928
-h, -help
  command line option, 890
-h
  genConfigRoutes.py command line option, 893
toaccess command line option, 928
-timetoaccess command line option, 928
r N, -num-retries N
  command line option, 890
-r PATH, -results_path PATH
  compare command line option, 893
-r PATH
  toaccess command line option, 928
-s, -snapshot
genConfigRoutes.py command line option, 894
-s, -traffic-ops-insecure
  command line option, 890
-s
  traffic_ops_ort command line option, 248
-t TIMEOUT,  
  -traffic-ops-timeout-milliseconds
TIMEOUT
  command line option, 890
-t
  traffic_ops_ort command line option, 246
-u TO_URL, -traffic-ops-url
TO_URL
  command line option, 890
-u TO_URL
  traffic_ops_ort command line option, 246
-v, -version
  command line option, 890
-v
  genConfigRoutes.py command line option, 894
toaccess command line option, 246
-w WARNING_LOCATION,
Apache Traffic Control Documentation, Release 4.0.0

A
add_ssl_keys_to_deliveryservice() (trafficops.TOSession method), 898
admin command line option
- cfg CONFIG, 233
- env ENVIRONMENT, 230
- fixtures FIXTURES, 233
command, 230
ALL (traffic_ops_ort.configuration.LogLevels attribute), 252
API (class in traffic_ops_ort.to_api), 256
api (traffic_ops_ort.configuration.Configuration attribute), 251
api_base_url (trafficops.RestApiSession attribute), 922
API_LOGGERS (in module traffic_ops_ort.to_api), 257
api_request() (in module trafficops), 922
api_version (trafficops.RestApiSession attribute), 923
assign_delivery_services_to_federations() (trafficops.TOSession method), 898
assign_deliveryservice_servers_by_ids() (trafficops.TOSession method), 898
assign_deliveryservice_servers_by_names() (trafficops.TOSession method), 899
assign_federation_resolver_to_federations() (trafficops.TOSession method), 899
assign_parameter_to_profile_ids() (trafficops.TOSession method), 899
assign_profile_to_parameter_ids() (trafficops.TOSession method), 899
associate_paramater_to_profile() (trafficops.TOSession method), 899
associate_parameters_by_profile_id() (trafficops.TOSession method), 899
associate_parameters_by_profile_name() (trafficops.TOSession method), 899
astats (stats_over_http), 935
base_url (trafficops.TOSession attribute), 899

C
Cache Group, 935
Cache Groups, 935
cache server, 936
cache servers, 936
CACHED_STATUSES (in module traffic_ops_ort.to_api), 257
cachegroups_queue_update() (trafficops.TOSession method), 900
CDN Snapshot, 946
CDN Snapshots, 946
cdnId (traffic_ops_ort.to_api.ServerInfo attribute), 258
cdnName (traffic_ops_ort.to_api.ServerInfo attribute), 258
cdns_queue_update() (trafficops.TOSession method), 900
checkInstallList() (traffic_ops_ort.packaging.Package method), 255
close() (trafficops.RestApiSession method), 923
command() (trafficops.TOSession method), 898
admin command line option
- dir directory, 184
- http port, 184
- started filename, 184
- P TO_PASSWORD,
- traffic-ops-password TO_PASSWORD, 890
- U TO_USER,
- traffic-ops-user TO_USER, 890
- a AGE,
- cache-file-max-age-seconds AGE, 890
- ERROR_LOCATION,
- log-location-error ERROR_LOCATION, 890
- g, -print-generated-files, 890
- h, -help, 890
- I INFO_LOCATION,
-log-location-info
INFO_LOCATION, 890
-l, -list-plugins, 890
-n, -no-cache, 890
-r N, -num-retries N, 890
-s, -traffic-ops-insecure, 890
-t TIMEOUT,
-traffic-ops-timeout-milliseconds TIMEOUT, 890
-u TO_URL, -traffic-ops-url TO_URL, 890
-v, -version, 890
-w WARNING_LOCATION,
-log-location-warning WARNING_LOCATION, 891

compare command line option
-ref_passwd PASSWD, 892
-ref_url URL, 892
-ref_user USER, 892
-test_password PASSWD, 892
-test_url URL, 892
-test_user USER, 892
-V, -version, 893
-f FILE, -file FILE, 892
-h, -help, 893
-r PATH, -results_path PATH, 893

Configuration (class in traffic_ops_ort.configuration), 251
Configuration.Modes (class in traffic_ops_ort.configuration), 251
content routing, 936

copy_profile() (trafficops.TOSession method), 900

Coverage Zone File, 936
Coverage Zone Map, 936

create() (trafficops.RestApiSession method), 923
create_asn() (trafficops.TOSession method), 900
create_cache_group_fallbacks() (trafficops.TOSession method), 900
create_cachegroups() (trafficops.TOSession method), 900
create_capability() (trafficops.TOSession method), 900
create_cdn() (trafficops.TOSession method), 900
create_cdn_dns_sec_keys() (trafficops.TOSession method), 901
create_coordinates() (trafficops.TOSession method), 901
create_delivery_service_user_link() (trafficops.TOSession method), 901
create_deliveryservice() (trafficops.TOSession method), 901
create_deliveryservice_regexes() (trafficops.TOSession method), 901
create_division() (trafficops.TOSession method), 901
create_federation() (trafficops.TOSession method), 901
create_federation_in_cdn() (trafficops.TOSession method), 901
create_federation_resolver() (trafficops.TOSession method), 902
create_federation_user() (trafficops.TOSession method), 902
create_invalidation_job() (trafficops.TOSession method), 902
create_origins() (trafficops.TOSession method), 902
create_parameter() (trafficops.TOSession method), 902
create_physical_location() (trafficops.TOSession method), 902
create_profile() (trafficops.TOSession method), 902
create_region() (trafficops.TOSession method), 902
create_server() (trafficops.TOSession method), 903
create_servercheck() (trafficops.TOSession method), 903
create_staticdnsentries() (trafficops.TOSession method), 903
create_tenant() (trafficops.TOSession method), 903
create_to_extension() (trafficops.TOSession method), 903
create_ts_databases command line option
-help, 162
-password password, 162
-replication N, 162
-url URL, 162
-user username, 162
create_user() (trafficops.TOSession method), 903
create_user_with_registration() (trafficops.TOSession method), 903
CRITICAL (traffic_ops_ort.configuration.LogLevels attribute), 252

D
DATA
toaccess command line option, 928
DEBUG (traffic_ops_ort.configuration.LogLevels attribute), 252
Deep Coverage Zone File, 937
Deep Coverage Zone Map, 937
delete() (in module to_access), 930
delete() (trafficops.RestApiSession method), 923
delete_asn() (trafficops.TOSession method), 903
delete_cache_group_fallbacks() (trafficops.TOSession method), 903
delete_cache_group_parameters() (trafficops.TOSession method), 904
delete_cachegroups() (trafficops.TOSession method), 904
delete_cdn_by_id() (trafficops.TOSession method), 904
delete_cdn_dns_sec_keys() (trafficops.TOSession method), 904
delete_coordinates() (trafficops.TOSession method), 904
delete_deliveryservice_user_link() (trafficops.TOSession method), 904
delete_deliveryservice_by_id() (trafficops.TOSession method), 904
delete_deliveryservice_regex_by_regex_id() (trafficops.TOSession method), 905
delete_deliveryservice_servers_by_id() (trafficops.TOSession method), 905
delete_deliveryservice_ssl_keys_by_xml_id() (trafficops.TOSession method), 905
delete_division() (trafficops.TOSession method), 905
delete_federation_in_cdn() (trafficops.TOSession method), 905
delete_federation_resolver() (trafficops.TOSession method), 905
delete_federation_user() (trafficops.TOSession method), 905
delete_origins() (trafficops.TOSession method), 906
delete_parameter() (trafficops.TOSession method), 906
delete_physical_location() (trafficops.TOSession method), 906
delete_profile_by_id() (trafficops.TOSession method), 906
delete_profile_parameter_association_by_id() (trafficops.TOSession method), 906
delete_region() (trafficops.TOSession method), 906
delete_server_by_id() (trafficops.TOSession method), 906
delete_staticdnsentries() (trafficops.TOSession method), 906
deleteOldStatusFiles() (in module traffic_ops_ort.main_routines), 253
Delivery Service, 937
Delivery Service required capabilities, 938
Delivery Services, 937
DISTRO (in module traffic_ops_ort.configuration), 252
Division, 938
Divisions, 938
doMain() (in module traffic_ops_ort), 250

E
Edge, 938
Edge-tier, 938
Edge-tier cache, 938
Edge-tier cache server, 938
Edge-tier cache servers, 938
Edge-tier caches, 938
environment variable MOJO_MODE, 84, 96, 97, 99, 230
SESSION_TIMEOUT_IN_SECS, 233, 236
TEST_PASSWORD, 892, 894, 895
TEST_URL, 892, 893, 895
TEST_USER, 892, 894, 895
TO_PASS, 890, 891
TO_PASSWORD, 245, 248, 892–895, 928, 929
TO_URL, 234, 236, 245, 247, 248, 890–894, 928, 929
TO_USER, 245, 248, 890–895, 929
TO_USER_ADMIN, 234, 237
TO_USER_EXTENSION, 234, 237
TO_USER_FEDERATION, 235, 237
TO_USER_OPERATIONS, 235, 237
TO_USER_PASSWORD, 245, 248, 890–895, 929
TO_USER_PORTAL, 235, 237
TO_USER_READ_ONLY, 235, 237
TODB_DESCRIPTION, 233, 238
TODB_HOSTNAME, 234, 238
TODB_NAME, 234, 238
TODB_PASSWORD, 234, 238
TODB_PORT, 234, 238
TODB_SSL, 234, 238
TODB_TYPE, 234, 238
TODB_USER, 234, 239

ERROR

(fic_ops_ort.configuration.LogLevels
attribute), 252

F

FATAL

(fic_ops_ort.configuration.LogLevels
attribute), 252

Federation, 938
Federations, 938
forward proxies, 939
forward proxy, 939
fullHostname

(fic_ops_ort.configuration.Configuration
attribute), 251

G

genConfigRoutes.py command line option
-refPasswd PASSWD, 893
-refURL URL, 893
-refUser USER, 893
-testPasswd PASSWD, 894
-testURL URL, 893
-testUser USER, 893
-C, -no-server-configs, 894
-h, -help, 893
-k, -insecure, 894
-l LOG_LEVEL, -log_level
LOG_LEVEL, 894
-s, -snapshot, 894
-v, -version, 894

generate_deliveryservice_ssl_keys()
(trafficops.TOSession method), 907

generate_deliveryservice_url_signature_keys()
(trafficops.TOSession method), 907

generate_iso()  (trafficops.TOSession
method), 907

generate_deliveryservice_ssl_keys()
(trafficops.TOSession method), 907

generate_deliveryservice_url_signature_keys()
(trafficops.TOSession method), 907

generate_iso()  (trafficops.TOSession
method), 907

generate_deliveryservice_ssl_keys()
(trafficops.TOSession method), 907

generate_deliveryservice_url_signature_keys()
(trafficops.TOSession method), 907

generate_iso()  (trafficops.TOSession
method), 907

get() (in module to_access), 930

get()  (trafficops.RestApiSession
method), 923

get_all_cachegroup_parameters()
(trafficops.TOSession method), 907

get_all_deliveryservice_servers()
(trafficops.TOSession method), 907

get_api_capabilities()  (trafficops.TOSession
method), 907

get_api_capabilities_by_id()
(trafficops.TOSession method), 907

get_asn_by_id()  (trafficops.TOSession
method), 907

get_asns()  (trafficops.TOSession
method), 907

get_associated_profiles_by_parameter_id()
(trafficops.TOSession method), 907

get_authenticated_user()  (trafficops.TOSession
method), 908

get_authenticated_user_jobs()
(trafficops.TOSession method), 908

get_cache_group_fallbacks()
(trafficops.TOSession method), 908

get_cache_stats()
(trafficops.TOSession method), 908

get_cachegroup_by_id()  (trafficops.TOSession
method), 908

get_cachegroup_parameters()
(trafficops.TOSession method), 908

generate_deliveryservice_ssl_keys()
(trafficops.TOSession method), 907

generate_deliveryservice_url_signature_keys()
(trafficops.TOSession method), 907

generate_iso()  (trafficops.TOSession
method), 907

get_cachegroup_unassigned_parameters()
get_origins()  
get_osversions()  
get_parameter_by_id()  
get_parameters()  
get_parameters_by_profile_id()  
get_parameters_by_profile_name()  
get_pending_snapshot_crconfig()  
get_physical_location_by_id()  
get_physical_locations()  
get_profile_by_id()  
get_profile_specific_config_files()  
get_profiles()  
get_region_by_id()  
get_regions()  
get_roles()  
get_server_by_id()  
get_server_config_files()  
get_server_delivery_services()  
get_server_details()  
get_server_specific_config_file()  
get_server_status_count()  
get_server_type_count()  
get_server_update_status()  
get_servers()  
get_static_dns_entries()  
get_staticdnsentries()  
get_statuses()  
get_statuses_by_id()  
get_system_info()  
get_tenant_by_id()  
get_tenants()  
get_to_extensions()  
get_unassigned_profiles_by_parameter_id()  
get_unnassigned_parameters_by_profile_id()  
get_user_by_id()  
get_user_delivery_services()  
get_unassigned_parameters_by_profile_id()  
get_unnassigned_profiles_by_parameter_id()  
get_types()  
get_types_only_names()  
getJSONResponse()  
getMyChkconfig()  
getMyConfigFiles()  
getMyPackages()
fic_ops_ort.to_api.API (method), log_with_debug_info() (in module trafficops.utils), 926
getMyStatus() (traffic_ops_ort.to_api.API method), logged_in (trafficops.TOSession attribute), 918
getMyUpdateStatus() (traffic_ops_ort.to_api.API method), traffic_ops_ort command line option, 247
getRaw() (traffic_ops_ort.to_api.API method), login() (trafficops.TOSession method), 918
getTextResponse() (in module traffic_ops_ort.utils), LoginError, 922
getYesNoResponse() (in module traffic_ops_ort.utils), LogLevels (class in traffic_ops_ort.configuration), 252

H
head() (in module to_access), 930
head() (trafficops.RestApiSession method), 924
Health Protocol, 941

I
INFO (traffic_ops_ort.configuration.LogLevels attribute), 252
install() (traffic_ops_ort.packaging.Package method), 255
installArgs (traffic_ops_ort.packaging.Package attribute), 255
INTERACTIVE (traffic_ops_ort.configuration.Configuration attribute), 251
InvalidJSONError, 921
is_open (trafficops.RestApiSession attribute), 924
isInstalled() (traffic_ops_ort.packaging.Package method), 255

L
localization, 941
LOG_FORMAT (in module traffic_ops_ort.configuration), 252
LOG_LEVEL, -1 LOG_LEVEL, -log_level LOG_LEVEL traffic_ops_ort command line option, 247

M
main() (in module traffic_ops_ort), 250
Mid, 941
Mid-tier, 941
Mid-tier cache, 941
Mid-tier cache server, 941
Mid-tier cache servers, 941
Mid-tier caches, 941
MODE
traffic_ops_ort command line option, 246
MOJO_MODE, 96, 97, 99

N
ame (traffic_ops_ort.packaging.Package attribute), 256
NONE (traffic_ops_ort.configuration.LogLevels attribute), 252

Q
Modes
OperationError, 922
options() (in module to_access), 930
options() (trafficops.RestApiSession method), 924
origin, 941
origin server, 941
origin servers, 941
origins, 941
ORT, 941
ORTException, 253
output() (in module to_access), 930

P
Package (class in traffic_ops_ort.packaging), 255
Parameter, 941
Parameters, 941
Index 959

parent, 941
parents, 941
parse_arguments() (in module to_access), 930
parseTOURL() (in module traffic_ops_ort.configuration), 252
parseTSRoot() (in module traffic_ops_ort.configuration), 253
patch() (in module to_access), 931
patch() (trafficops.RestApiSession method), 925
PATH
toaccess command line option, 927
Physical Location, 941
Physical Locations, 941
post() (in module to_access), 931
post() (trafficops.RestApiSession method), 925
processConfigurationFiles() (in module traffic_ops_ort.main_routines), 254
processPackages() (in module traffic_ops_ort.main_routines), 254
processServices() (in module traffic_ops_ort.main_routines), 254
Profile, 941
profileId (traffic_ops_ort.to_api.ServerInfo attribute), 258
profileName (traffic_ops_ort.to_api.ServerInfo attribute), 258
Profiles, 942
put() (in module to_access), 931
put() (trafficops.RestApiSession method), 925
Q
Queue, 942
Queue Server Updates, 942
Queue Updates, 942
R
Region, 943
Regions, 943
replace_authenticated_user() (trafficops.TOSession method), 918
REPORT (traffic_ops_ort.configuration.Configuration.Modes attribute), 251
request() (in module to_access), 931
resp (trafficops.InvalidJSONError attribute), 922
resp (trafficops.OperationError attribute), 922
RestApiSession (class in trafficops), 922
REVALIDATE (traffic_ops_ort.configuration.Configuration.Modes attribute), 251
revalidateState() (in module traffic_ops_ort.main_routines), 254
reverse proxies, 943
reverse proxy, 943
RFC
RFC 3339, 326
RFC
RFC 1035, 35, 426, 432, 442, 448
RFC 2616, 4, 191, 943
RFC 2616#section-14.9, 5
RFC 3339, 327, 328, 406, 472–474, 481, 490, 496, 500, 524, 563, 612, 615, 622, 770, 798–800, 849, 850, 876
RFC 4033, 141
RFC 4044, 141
RFC 4045, 141
RFC 4122, 858
RFC 4511, 98
RFC 6781, 141
RFC 6781#section-4.1.1.1, 142
RFC 7129, 141
RFC 7516, 558–560
RFC 7518, 558–560
Role, 945
Roles, 945
run() (in module traffic_ops_ort.main_routines), 254
S
sanitize() (traffic_ops_ort.to_api.ServerInfo method), 258
Server Capabilities, 945
Server Capability, 945
server_url (trafficops.RestApiSession attribute), 926
serverId (trafficops.or.to_api.ServerInfo attribute), 258
ServerInfo (class in trafficops.or.to_api), 258
ServerInfo (trafficops.or.configuration.Configuration attribute), 251
serverIpv4 (trafficops.or.to_api.ServerInfo attribute), 258
serverName (trafficops.or.to_api.ServerInfo attribute), 258
servers_queue_update() (trafficops.TOSession method), 918
serverTcpPort (trafficops.or.to_api.ServerInfo attribute), 258
session (trafficops.RestApiSession attribute), 926
SESSION_TIMEOUT_IN_SECS, 236
setLogLevel() (in module trafficops.or.configuration), 253
setStatusFile() (in module trafficops.or.main_routines), 254
shortHostname (trafficops.or.configuration.Configuration attribute), 252
Snapshot, 945
snapshot_crconfig() (trafficops.TOSession method), 919
Snapshots, 946
Status, 946
STATUS_FILE_DIR (in module trafficops.or.main_routines), 253
Statuses, 946
csync_ts_databases command line option

-T, Tenancies, 946
-Tenancy, 946
-Tenant, 946
-Tenants, 946
-TEST_PASSWORD, 892, 894
-TEST_URL, 892, 893
-TEST_USER, 892, 894
to_access (module), 927

-TO_PASS, 890
to_url (trafficops.TOSession attribute), 919
to_url (trafficops.TOSession attribute), 919
to_url (trafficops.TOSession attribute), 919
to_url (trafficops.TOSession attribute), 919

-TO_PASSWORD, 245, 248, 892–894, 928
-TO_PASSWORD, 245, 248, 892–894, 928

-TO_URL, 236, 245, 247, 248, 890, 892, 893, 928, 929

-TO_URL, 236, 245, 247, 248, 890, 892, 893, 928, 929

-TO_USER, 245, 248, 890, 892–894, 929

-TO_USER_ADMIN, 237
-TO_USER_DISALLOWED, 237
-TO_USER_EXTENSION, 237
-TO_USER_FEDERATION, 237
-TO_USER_OPERATIONS, 237

-TO_USER_PASSWORD, 236
-TO_USER_PORTAL, 237

-TO_USER_READ_ONLY, 237

-toaccess command line option

-request-headers, 928
-request-payload, 928
-response-headers, 928

-to-password PASSWORD, 928
-to-url URL, 928
-to-user USERNAME, 928

-a API_VERSION,

-api-version API_VERSION, 928

-f, -full, 928
-h, -help, 928
-k, -insecure, 928
-p, -pretty, 928
-r, -raw-path, 928
DATA, 928
PATH, 928
TODB_DESCRIPTION, 238
TODB_HOSTNAME, 238
TODB_NAME, 238
TODB_PASSWORD, 238
TODB_PORT, 238
TODB_SSL, 238
TODB_TYPE, 238
TODB_USER, 239
toRevProxyUrl (traffic_ops_ort.to_api.ServerInfo attribute), 258
TOSession (class in trafficops), 897
TOURL (traffic_ops_ort.configuration.Configuration attribute), 251
toUrl (traffic_ops_ort.to_api.ServerInfo attribute), 258
TRACE (traffic_ops_ort.configuration.LogLevels attribute), 252
traffic_ops command line option
-api-routes, 84
-cfg CONFIG_PATH, 84
-dbcfg DB_CONFIG_PATH, 84
-plugins, 84
-riakcfg
 TRAFFIC_VAULT_CONFIG_PATH, 85
-version, 85
traffic_ops_ort (module), 245
traffic_ops_ort command line option
-dispersion DISP, 246
-login Dispersion DISP, 246
-retries RETRIES, 246
-rev_prox_disable, 246
-ts_root PATH, 246
-wait_for_parents INT, 246
-h, -help, 246
-k, -insecure, 246
-p PASSWORD, -to_password PASSWORD, 248
-u USER, -to_user USER, 248
-v, -version, 246
LOG_LEVEL, -l LOG_LEVEL,
-log_level LOG_LEVEL, 247
LOGIN, 247
MODE, 246
TO_URL, -to_url TO_URL, 247
traffic_ops_ort.configuration (module), 251
traffic_ops_ort.main_routines (module), 253
traffic_ops_ort.packaging (module), 255
traffic_ops_ort.to_api (module), 256
traffic_ops_ort.utils (module), 259
traffic_vault_util command line option
-dry_run, 931
-vault_action ACTION, 931
-vault_ip IP, 931
-vault_password PASSWD, 932
-vault_port PORT, 932
-vault_user USER, 932
trafficops (module), 897
trafficops.__version__ (module), 927
trafficops.utils (module), 926
Type, 946
Types, 946
U
uninstall() (traffic_ops_ort.packaging.Package method), 256
uninstallArgs (traffic_ops_ort.packaging.Package attribute), 256
update_asn() (trafficops.TOSession method), 919
update_cache_group_fallbacks() (trafficops.TOSession method), 919
update_cachegroups() (trafficops.TOSession method), 919
update_cdn_by_id() (trafficops.TOSession method), 919
update_coordinates() (trafficops.TOSession method), 919
update_deliveryservice_by_id() (trafficops.TOSession method), 919
update_deliveryservice_regexes() (trafficops.TOSession method), 920
update_deliveryservice_safe() (trafficops.TOSession method), 920
update_division() (trafficops.TOSession method), 920
update_federation_in_cdn() (trafficops.TOSession method), 920
update_origins() (trafficops.TOSession method), 920
update_parameter() (trafficops.TOSession method), 920
update_physical_location() (trafficops.TOSession method), 920
update_profile_by_id() (trafficops.TOSession method), 921
update_region() (trafficops.TOSession method), 921
update_server_by_id() (trafficops.TOSession method), 921
update_server_status_by_id() (trafficops.TOSession method), 921
update_staticdnentries() (trafficops.TOSession method), 921
update_tenant() (trafficops.TOSession method), 921
update_user_by_id() (trafficops.TOSession method), 921
updateTrafficOps() (traffic_ops_ort.to_api.API method), 257

V
validate_parameter_exists() (trafficops.TOSession method), 921
version (traffic_ops_ort.packaging.Package attribute), 256
VERSION (traffic_ops_ort.to_api.API attribute), 256

W
WARN (traffic_ops_ort.configuration.LogLevels attribute), 252