Stingray™ Traffic Manager
REST API Guide

Version 9.5

December 2013

riverbed®

Riverbed Technology
199 Fremont Street
San Francisco, CA 94105

Phone: 415.247.8800
Fax: 415.247.8801
Web: http://www.riverbed.com

Part Number
712-00158-02

Contents

CHAPTER 1 Introduction 6

Introducing Stingray 6
Introducing REST 6
Why use a REST API 7
A REST-Based Architecture 7
Scope of This Release 8

CHAPTER 2 Typical Usage in Stingray 9

The Resource Model 9
Sections 9
Data Types 10
Resource URI Patterns 12
Traversing the Tree 13
The Traffic Manager REST Service 15
The Application Firewall REST Interface 15
Authentication 16
Supported HTTP Methods 16
Requesting a Resource 17
Setting Configuration for a Resource 17
Removing Resources 18
Further Aspects of the Resource Model 18
Enumerated Types 18
Uploading Files 18
Custom Configuration Sets 18
Errors 19
Stingray UI Features 20
Enabling and Disabling the API 20
Controlling Timeout Events 20
Configuring the IP Addresses That the REST API Listens On 21
Restricting Access to Trusted Users 21
Log Messages in Stingray 22

CHAPTER 3 Examples and Use-Cases 24

Typical Usage 24
Listing Running Virtual Servers 25
Adding a Node to a Pool 27

CHAPTER 4 Resource Model Reference 29

About the Resource Model Reference 29
Configuration Resources 29
Action Program 29
Alerting Action 29

Aptimizer Application Scope 34
Aptimizer Profile 34
Bandwidth Class 36
Cloud Credentials 36
Custom configuration set 38
Event Type 38
Extra File 43
GLB Service 43
Global Settings 47
License 78
Location 79
Monitor 80
Monitor Program 85
Pool 85
Protection Class 99
Rate Shaping Class 102
Rule 103
SLM Class 103
SSL Client Key Pair 104
SSL Key Pair 105
SSL Trusted Certificate 105
Security Settings 106
Session Persistence Class 106
Traffic IP Group 108
Traffic Manager 110
TrafficScript Authenticator 120
User Authenticator 122
User Group 128
Virtual Server 129
SNMP Counter Resources 152
Actions 152
Asp session cache 152
Bandwidth 154
Cloud api credentials 155
Connection rate limit 155
Events 157
Glb services 157
Globals 158
Ip gateway 170
Ip session cache 172
J2ee session cache 173
Listen ips 175
Locations 176
Network interface 177
Node 179
Node inet46 181
Per location service 184
Per node service level 186
Per node service level inet46 187
Per pool node 188
Pools 191

Rule authenticators 195
Rules 196
Service level monitors 197
Service protection 199
Ssl cache 200
Ssl ocsp stapling 202
Ssl session cache 203
Traffic ip 204
Traffic ip inet46 205
Uni session cache 206
Virtual servers 207
Web cache 213
System Information Resources 216
Information 216

CHAPTER 5 Further Information and Resources 217

Stingray Manuals 217
Information Online 217

APPENDIX A Changes in this Version of the API 218

CHAPTER 1 Introduction

Introducing Stingray

The Stingray product family provides high-availability, application-centric traffic management and load balancing solutions. They provide control, intelligence, security and resilience for all your application traffic.

Stingray products are intended for organizations hosting valuable business-critical services, such as TCP and UDP-based services like HTTP (web) and media delivery, and XML-based services such as web services.

Introducing REST

REST (REpresentational State Transfer) is a framework for API design. It is based on generic facilities of the standard HTTP protocol, including the six basic HTTP methods (GET, POST, PUT, DELETE, HEAD, INFO) and the full range of HTTP return codes.

A REST interface partitions the API into a series of "resources", each of which can be accessed using one or more HTTP methods. (In Stingray, only the GET, PUT, and DELETE methods are used; HEAD, POST and INFO are not currently implemented). Each method operates in Stingray as follows:

- GET: Obtain a representation of the resource, without modifying server state (except perhaps for logging purposes).
- PUT: Create a new resource or apply some change to a resource. Where the resource exists, only those properties specified in the request are modified; all others remain unchanged. If a resource object does not exist, a new one is created.
- DELETE: Delete an existing resource.

Importantly, each resource is uniquely identified with an address, or URI (Uniform Resource Identifier). In other words, if you know the URI you can access the resource (subject to the normal authorization/authentication processes associated with accessing the administrative systems of the Traffic Manager).

Since all resources have URIs, resources can point to other resources by embedding the URIs of related resources within their representations.

In Stingray, all resources are represented and stored as JSON (JavaScript Object Notation) structures. Requests and responses that interact with the Traffic Manager through the REST API must adopt the same format.

The full range of HTTP return codes is available in REST, although in practise a useful subset can be identified and applied consistently. So, for example, it should be evident from the response itself whether a request has succeeded or not, without any need for parsing the body of the response. However, Stingray always attempts to provide extra information regarding a failure into the response body. Refer to the "Errors" section of CHAPTER 2 for more details.

Why use a REST API

REST interfaces have become popular in public APIs because of their inherent simplicity. An API can focus on available resources, with details regarding updating and deleting of each resource delegated to the appropriate HTTP method in predictable ways.

The purpose of implementing a REST API is not primarily to add functionality but to add structure. Because of the inherent similarity of all REST APIs (by virtue of their underlying HTTP structure), familiarity with any REST API brings familiarity with all of them. In many cases it is just as easy to implement to a REST design as it is to use a more ad hoc API design, while reaping the benefits that come with well-understood REST conventions.

Finally, the availability of return codes is another example of leveraging known semantics when building a useful API. Without a meaningful return code it becomes necessary to parse every response to find out whether it worked or not. In addition, most modern browsers and Web programming frameworks expect that specific HTTP error codes are set in the event of error and respond differently depending on the code. This is especially apparent in the case of AJAX requests, which are often handled differently by many modern Javascript frameworks depending on the status code returned from the server.

A REST-Based Architecture

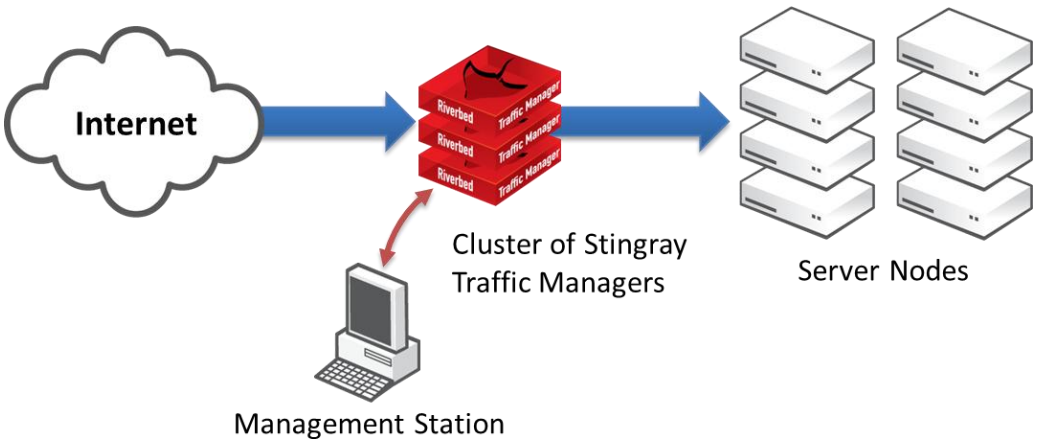

Fig. 1. Arrangement of Management Server, Stingray Cluster and Server Nodes

A cluster of traffic mangers is normally managed using the web-based Administration UI on one of the machines. Stingray's REST API provides an alternative means to remotely administer and configure a Stingray cluster.

Stingray's REST service is disabled by default, and must first be enabled from the Administration UI before it can be used. See "Enabling and Disabling the API" in CHAPTER 2 for more details.

The Stingray REST API can be used by any HTTP client or application environment that supports HTTP services.

The REST API is an interface used to configure, manage, and monitor a cluster of Traffic Managers remotely.

A management application can issue a REST request to one of the Traffic Managers in a Stingray cluster. The application may be running on a stand-alone management server, one of the server nodes, or even on one of the Traffic Managers.

The application can issue the request to any of the Stingray Traffic Managers. The Traffic Managers automatically synchronize their configuration, so a configuration change sent to one machine is automatically replicated across the cluster.

Important: Due to the nature of the REST API's ability to access and modify your Traffic Manager configuration, it is strongly recommended that you disallow access to this service from outside of your trusted network.

Scope of This Release

This document describes the features and capabilities of the REST API for the Stingray Traffic Manager 9.5 release. The REST API version referred to in this document is 2.0.

Basic type-checking is performed by the API, however you should ensure that your client application provides suitable validation to ensure the suitability of the configuration data being provided to the Traffic Manager.

All defined users in the system have the ability to authenticate a connection through the Traffic Manager REST API. However, you cannot modify the users configuration file in any way, so it is not possible to add, edit, or delete users through the API.

A full list of specific features, capabilities, and API versions supported by this release can be found in the release notes supplied with your product variant.

CHAPTER 2 Typical Usage in Stingray

The Resource Model

The Stingray REST API is made up of a hierarchy of resources that are manipulated using standard HTTP calls to a listener service running along side the Traffic Manager. HTTP URIs are used to address the resources in the system.

There are three resource *types*:

- **Configuration** – to represent Traffic Manager configuration objects.
- **Counters** – for reporting through SNMP counters.
- **Information** – for system information.

"Counters" and "Information" resources are read-only, whereas "configuration" resources are fully interactive and map directly to the native Stingray configuration system. Each concept, such as pools, virtual servers, TrafficScript rules, or Service Level Monitoring classes, has an associated configuration resource model.

All resources are represented as JSON structures (MIME type `application/json`), and objects of each resource type are captured in this format.

Typically, a configuration resource follows this format:

```
{
   "properties": {
      "sectionname": {
         "key1": "stringvalue1",
         "key2": numericvalue2,
         "key3": booleanvalue3
      }
   }
}
```

A single instance of a resource, for example a virtual server, contains a primary group entitled "**properties**". This contains all configuration keys attributable to this resource type.

Counter resources contain dynamically generated data to correspond to SNMP counters in the Traffic Manager.

Sections

Sections are designed to contain properties (or "keys") that have a commonality of purpose or perhaps apply in certain circumstances. For example, monitor classes may have keys that apply only to monitors of particular types.

In a configuration resource, the properties group contains several sections, one for each *logical set* of keys. There is always a section entitled "**basic**," containing common configuration items, followed by one or more additional sections according to the specification of the resource.

A counter resource contains a single section, "**statistics**," listing the SNMP counters associated with the resource. Similarly, an information resource contains a single section, "**information**," listing the system information properties applicable to this Traffic Manager.

Data Types

Each `key:value` pair is then presented as a comma-separated list within each section, according to the specification shown throughout this guide. Key names are always delimited by quotes, with the values according to the following rules:

Boolean	A value of `true` or `false` (case-sensitive). For example:
	`"key1": true,`
	`"key2": false`

Int	A numeric positive or negative value with no decimal point. For example:
	`"key1": 1024,`
	`"key2": -10`

Unsigned Int	A numeric positive value with no decimal point. For example:
	`"key1": 0,`
	`"key2": 50`

Float	A numeric positive or negative value that can have a decimal point. For example:
	`"key1": 1.0,`
	`"key2": -1024.111`

String	A set of alpha-numeric characters that may not include new-lines. Non-alpha characters must use correct character escapes. For example:
	`"key1": "Hello world",`
	`"key2": "",`
	`"key3": "Hello y\'all"`

Freeform String	A set of alpha-numeric characters that can contain new-lines. Non-alpha characters must use correct character escapes, and a newline must be represented by a `\n`. For example: `"key1": "Multi-line\nString",`
Password	A string that cannot be read, only written to. When read, it is displayed as a structure that indicates if the password has been set (is non-empty). For example, when reading the key:

	"key1": { "password_set": false }, "key2": { "password_set": true } When writing to the key, the structure can be unchanged, or a new password can be set: "key2": { "password_set": true }, "key1": "secret123"
Time	Times are represented as strings in ISO8601 time format, including a time zone designator. For example: {Year}-{Month}-{Day}T{Hour}:{Minute}:{Second}{Time Zone}
Set	This is a collection of unique un-ordered items of a particular type, stored as an array. For consistency, a set is rendered in alpha-numeric order. For example: "key": ["Item A", "Item B", "Item D"]
List	This is a collection of ordered items of a particular type. It may contain duplicates and is stored as a standard array. For example: "key": ["Item A", "Item C", "Item A"]
Tables	This is a special type designed to allow nested data within a single config key. In some circumstances, you might wish to specify a list/array of data items, such as a list of pool nodes, where each item has one or more extra pieces of configuration data to be attached to it. Each one of these nested list entries expects a value known as the **primary key**, used to identify it. Each sub-key value should then be specified in the same way. For example: "key": [{ "prmkey": "Hello World", "subkey1": false, "subkey2": ["Item 1", "Item 2"] }, { "prmkey": "Other text", "subkey1": true, "subkey2": [] },]

Resource URI Patterns

All Stingray resources are provided through a common base URI that identifies the root of the resource model. This is:

```
https://<host>:<port>/api/tm/<version>
```

In this URI path, `<host>` is the hostname of the Traffic Manager concerned, and `<port>` is the port that the REST API is published on (for example: https://myhost.example.com:9070). The `<version>` component refers to the REST version number. Add any supported REST version number here to access the API for that version. Supported versions are listed in the release notes supplied with your product variant.

Note: In the example above, a scheme of HTTPS is used to signify an encrypted connection from a remote client. HTTP is supported only where the connection is to a server on the same host. Refer to the Authentication section below for more details.

You can find different resource types at specific child nodes under this root URI:

- For configuration resources:

```
https://<host>:<port>/api/tm/<version>/config/active
```

- For counter resources:

```
https://<host>:<port>/api/tm/<version>/status/<host>/statistics
```

- For information resources:

```
https://<host>:<port>/api/tm/<version>/status/<host>/information
```

Instances of a particular configuration resource, such as a virtual server, are persistently stored and alter the host Traffic Manager's behaviour if changed. Additionally, changes made here are synchronized automatically to all other machines in the cluster.

Conversely, instances of counter and information resources are unique to each Traffic Manager in the cluster. Data for each cluster member is available by specifying the desired Traffic Manager hostname after the `/status` node in the URI.

Important: Resource URIs are case-sensitive.

If you wish to view (or modify, in the case of configuration resources) a stored record of a particular resource type, append the full path on to the end of the base URI. For example, a request for a virtual server configuration resource named "Web" would look like this:

```
https://myhost:9070/api/tm/2.0/config/active/virtual_servers/Web
```

Equally, a request for the SNMP counter data from host "myhost2", for a pool named "P1", would look like this:

```
https://myhost:9070/api/tm/2.0/status/myhost2/statistics/pools/P1
```

Traversing the Tree

Resource URIs can be either:

- resources, or
- a directory structure containing child elements denoting sub-directories or resource *nodes*.

You can test the overall availability of the REST API by querying the following URI:

```
https://<host>:<port>
```

(As mentioned above, `<host>` is the hostname of the Traffic Manager and `<port>` is the port that the REST API is published on).

A GET request for this URI should yield the following result:

```
{
   "children": [{
      "name": "api",
      "href": "/api/"
   }]
},
```

This shows that the REST service at `<host>:<port>` contains a single child element "`/api`". We know from the "Resource URI Patterns" section above that the full root URI of the configuration resource model is the following:

```
https://myhost:9070/api/tm/2.0/config/active
```

Therefore, requesting this URI results in a list of child elements similar to the following:

```
{
   "children": [{
      "name": "action_programs",
      "href": "/api/tm/2.0/config/active/action_programs/"
   }, {
      "name": "actions",
      "href": "/api/tm/2.0/config/active/actions/"
   }, {
      "name": "aptimizer",
      "href": "/api/tm/2.0/config/active/aptimizer/"
   }, {
      "name": "bandwidth",
      "href": "/api/tm/2.0/config/active/bandwidth/"
   }, {
```

```
      "name": "cloud_api_credentials",
      "href": "/api/tm/2.0/config/active/cloud_api_credentials/"
   }, {
   ...
   (truncated)
   ...
   }, {
      "name": "virtual_servers",
      "href": "/api/tm/2.0/config/active/virtual_servers/"
   }]
}
```

This output identifies all configuration resource types available through the Traffic Manager being queried. Each is identified by a `name` and `href` attribute combination.

A query for a specific resource type shows all instances of that resource defined within the Traffic Manager configuration. For example, the following URI lists all virtual servers:

```
https://myhost:9070/api/tm/2.0/config/active/virtual_servers
```

The output shows each stored virtual server, as per the following example:

```
{
   "children": [{
      "name": "vs1",
      "href": "/api/tm/2.0/config/active/virtual_servers/vs1"
   }, {
      "name": "vs2",
      "href": "/api/tm/2.0/config/active/virtual_servers/vs2"
   }]
}
```

SNMP counter and system information resources are unique to each Traffic Manager in the cluster. You can access the data for each cluster member from the API of whichever Traffic Manager you are connected to.

To list the available Traffic Managers in your cluster, perform a request for the following URI:

```
https://myhost1:9070/api/tm/2.0/status
```

The response is a list of child elements similar to the following:

```
{
   "children": [{
      "name": "myhost1.example.com",
      "href": "/api/tm/2.0/status/myhost1.example.com/"
   }, {
      "name": "myhost2.example.com",
      "href": "/api/tm/2.0/status/myhost2.example.com/"
   }, {
```

```
      "name": "myhost3.example.com",
      "href": "/api/tm/2.0/status/myhost3.example.com/"
   }, {
      "name": "local_tm",
      "href": "/api/tm/2.0/status/local_tm/"
   }]
}
```

The list also includes a `local_tm` child node that corresponds to the REST API of the Traffic Manager you are currently accessing. This provides a consistent programmatic interface to access resources for the local Traffic Manager only, no matter which host's API you are connected to. For example, the following URI can be used on the API of any Traffic Manager in the cluster, and the response contains results for that Traffic Manager only:

```
/api/tm/2.0/status/local_tm/information
```

To view (or modify, in the case of configuration resources) a stored record for a particular resource type, append the full path to the end of this base URI. For example, a request for a virtual server configuration resource named "Web" looks like this:

```
https://myhost:9070/api/tm/2.0/config/active/virtual_servers/Web
```

Equally, a request for the SNMP counter output for a pool named "P1" looks like this:

```
https://myhost:9070/api/tm/2.0/status/local_tm/statistics/pools/P1
```

The Traffic Manager REST Service

The Traffic Manager REST API is an HTTP service running on the Traffic Manager server. By default, it is available on TCP port **9070**, although this can be reconfigured. The REST service supports HTTP versions: 0.9, 1.0, and 1.1; Version 1.1 is recommended.

When connecting to the local machine using a loop-back interface (for example, 127.0.0.1 or "localhost"), plain HTTP must be used. When connecting from a remote machine, connections must be encrypted using SSL (HTTPS).

The service uses the same SSL certificate as the Traffic Manager's admin server, which by default is an automatically generated self-signed certificate. Any HTTP client used to connect to the REST API should have the server's self-signed certificate added to its trusted certificate catalogue. Alternatively the admin server/REST certificate can be replaced with one signed by a trusted certificate authority.

The Application Firewall REST Interface

The Stingray Application Firewall (SAF) component maintains a separate REST interface to facilitate control of SAF-specific resources. You can reach this interface through the standard Stingray REST service, using the following path:

```
https://<host>:<port>/api/af/<version>
```

`<version>` can be any specific currently published API version, or you can use the string "`latest`" to access the most current version. Full details of the available resources and actions that can be performed through the SAF REST API can be found in the SAF user documentation.

To access the SAF REST interface, you must first install and activate the SAF component on your Traffic Manager. You must also enable the Traffic Manager REST service through the Admin UI (see "Enabling and Disabling the API" below).

Stingray Traffic Manager operates as a proxy to the SAF REST service, and communicates with it through a designated port. You can view and modify this port from the *Stingray Application Firewall Ports* section on the **System > Application Firewall** page of the Admin UI. Any problems accessing the SAF REST interface can often be resolved by setting this value to a known free port. Contact your support provider if you require any further information.

Authentication

A REST-based management application communicates with a configuration service running on the Stingray *Admin Server* (the Traffic Manager-based service used to provide the Admin UI), so the same security considerations apply:

- REST requests are authenticated using HTTP Basic Auth.
- REST traffic over HTTPS is automatically encrypted using SSL. Traffic over HTTP is not encrypted, so should only be used inside a secure environment or to/from *localhost*.
- The Stingray Admin Server authenticates itself with its SSL certificate, which is generally self-signed. You may need to ensure that your REST application accepts self-signed certificates, or install a trusted SSL certificate in your Stingray admin server.
- REST requests are authenticated using the same user credentials as defined in the Administration Server. Individual object access is synonymous with page access in the Admin UI. For example, if a user wishes to view and manipulate pool objects, they must have been granted access to pools on the access permissions page.

Supported HTTP Methods

The REST service supports three primary HTTP methods for accessing and modifying data in the Stingray configuration system:

- GET
- PUT
- DELETE

GET is used when making read-only requests for a resource, whereas PUT is used when updating existing data or adding new configuration objects. DELETE is used when you wish to completely remove configuration objects from the Traffic Manager. Each of these is covered in more detail below.

Requesting a Resource

A client interacts with the Stingray REST API by performing operations on its resources. An operation is distinguished by the HTTP method used and the path and query components of the URI. Some operations, however, are not applicable to every resource.

The **GET** method is used to retrieve the current representation of the resource it is used on. It does not alter the resource in any way or have any other side effects.

This is achieved by sending a HTTP GET request to the server with no body. The request must accept a response in JSON format only (by specifying an **Accept** header type of `application/json`), and authorization is provided using **HTTP Basic Auth** (see the Authentication section above for more details). Such a request resembles the following:

```
GET /api/tm/2.0/config/active/bandwidth/BWClass1 HTTP/1.1
Authorization: Basic YWRtaW46c2VjcmV0MTIz
Accept: application/json
```

If successful, the server returns a "200 OK" response code with the full resource in the response body. The above *Bandwidth class* example might produce the following output:

```
{
   "properties": {
      "basic": {
         "maximum": 10000,
         "note": "This is my bandwidth class",
         "sharing": "cluster"
      }
   }
}
```

This is a JSON structure representing the configuration keys present in the requested bandwidth class object. In this case, it consists of a single "basic" section containing three key:value pairs. Other resource types might contain different or additional sections and corresponding keys.

Setting Configuration for a Resource

Note: This section does not apply to read-only resources such as SNMP counters or system information.

Changing data items in the Traffic Manager configuration system is achieved through a PUT request to a configuration type resource. This applies to either **creating** new resource items or **updating** the properties of an existing resource item.

When creating a new resource item, the request URI must contain the full path to the intended item, with the name being the final element of the path. For example, creating a new bandwidth class called "mynewclass" requires using the following URI:

```
/api/tm/2.0/config/active/bandwidth/mynewclass
```

For both creation and update operations, the request body must contain a representation of the resource properties in JSON format (with the appropriate body "Content-Type" header set). Partial updates to configuration resources can be performed by only including the properties that need to be altered. Other properties are left unchanged.

Note: For PUT requests, ensure that the request body is encoded as UTF-8.

The REST service returns a "200 OK" response for a correctly updated configuration set, or "201 Created" for establishing a new configuration object of a particular resource type. In these cases, the full resource is returned as the response body. The only exception to this rule is when updating a raw file, which instead returns a "204 No Content" empty-body response.

Important: You might want to exercise some care when creating or updating resources. The changes are permanent and no warning is given for existing configuration that is overridden. If you attempt to create a new resource where one of the same name already exists, you overwrite the properties of the existing record. It is recommended that you build such validation into your REST client application.

Removing Resources

Note: This section does not apply to read-only resources such as SNMP counters or system information.

A HTTP DELETE request for the full URI of a configuration item can be sent to the REST server to permanently remove it. On success, a "204 No Content" empty-body response is returned.

Further Aspects of the Resource Model

Enumerated Types

Some configuration keys can accept one or more of a pre-defined set of values. This is known as an enumerated key type, and the list of possible values (with long description) is provided in the reference guide later in this document.

Uploading Files

Resources that represent real files (such as TrafficScript rules) can also be presented in a raw format, where the data returned is the contents of the file. The MIME type of the request payload should be set to `application/octet-stream`.

Custom Configuration Sets

You can store and retrieve arbitrary *name:value* configuration pairs in the Traffic Manager configuration system using the REST API. This configuration is replicated across your cluster, and is only accessible through the REST API, SOAP API, and ZCLI.

To store a custom configuration, create an instance of the **custom** resource and set your *name:value* data to the **string_lists** property. For example, to create a resource called "customdata", use the following URI:

```
https://myhost:9070/api/tm/2.0/config/active/custom/customdata
```

Set the request body to a JSON structure resembling the following:

```
{
   "properties": {
      "basic": {
         "string_lists": [{
            "name": "customname1",
            "value": ["val1", "val2"]
         },{
            "name": "customname2",
            "value": "val3"
         }]
      }
   }
}
```

Using this system, you can organize your custom configuration into logical groups, initially by an instance of the **custom** resource, and within this, by a *name:value* pair. Each *value* can itself be a single item or a list of items.

Errors

If the REST server is unable to handle a HTTP request, it returns a HTTP response with an appropriate HTTP error code. The response body is in JSON and contains a data structure describing the error with a unique identifier (different than the numeric error code) and a description.

The unique identifier is made up of 2 parts:

```
{section}.{error_type}
```

Some errors might provide additional formatted information, specified with an optional "`error_info`" parameter. For example, the REST API uses this parameter to return per-property errors when a value fails validation. The following structure demonstrates the general form of an error:

```
{
  "error": {
     "error_id":   "{error identifier}",
     "error_text": "{error description}",
     "error_info": {error specific data structure, optional}
  }
}
```

A validation error occurs when one or more of the properties within a configuration resource fail a validation check. The `error_info` section then contains a sub-error for each property that failed validation. These sub-errors are like normal errors in that they contain an identifier (`error_id`) and a human readable text description (`error_text`):

```
{
  "error": {
     "error_id":   "resource.validation_failed",
     "error_text": "Some of the properties in the resource failed
validation.",
     "error_info": {
        "basic": {
           "key1": {
              "error_id": "num.range",
              "error_text": "Value must be in range 1000 - 2000."
           }
        }
     }
  }
}
```

Stingray UI Features

Enabling and Disabling the API

The REST service can be enabled or disabled from the *REST API* section of the **System > Security** page of the Stingray Admin UI. This page also provides the ability to set the TCP port that the service listens on. The default port is **9070**, however any unreserved port can be used here provided it does not conflict with other services already running on the Traffic Manager system. The changes are applied as soon as you click **Update**.

Important: The REST API is currently not available in conjunction with the Stingray Multi-Site Manager (MSM) feature. Attempts to enable the REST service whilst MSM is operational are denied. Equally, attempting to enable MSM whilst the REST service is running triggers an error. The current state of the Traffic Manager remains unchanged in either of these situations.

You can manually restart the REST API service from the **System > Traffic Managers** page. Click the *Restart REST API* button in the **Software Restart** section and confirm the restart on the next screen. Any existing connections are lost while the service restarts.

Controlling Timeout Events

The *REST API* section of the **System > Security** page provides a number of settings to control how the Traffic Manager responds to certain timeout events that occur through use of the REST API. These are:

`rest!auth_timeout`	The timeout period, in seconds, for the REST Authentication cache. As REST does not include the concept of a "session", each request must

	include user and password credentials. These credentials are validated each time; however, to save requesting repeated external authentications for the same user (from the same IP address), a cache of recent authentications is maintained. This timeout is the maximum amount of time a given user's credentials can stay in the cache. A setting of 0 (zero) disables the cache, forcing every REST request to be authenticated as it is received. However, this affects the performance of the API. (Default: 120 seconds)
`rest!replulltime`	This is the *lull time* for configuration replication via REST. This is the time, in seconds, of inactivity via the REST API before configuration replication starts. Increasing this value delays configuration replication among a cluster of Traffic Managers. (Default: 5 seconds)
`rest!repabstime`	This is the absolute timeout prior to configuration replication via REST. This is the longest time, in seconds, before configuration replication via REST starts, regardless of activity through the API. (Default: 20 seconds)
`rest!reptimeout`	The configuration replication duration timeout via REST. This is the time, in seconds, allowed for the process of configuration replication to complete. On a system with slow cluster communications or a very large configuration, increasing this value improves replication reliability. (Default: 10 seconds)

Configuring the IP Addresses That the REST API Listens On

The *REST API* section of the **System > Security** page contains a setting, `rest!bindips`, that can be used to control the IP address(es) that the REST API listens on for connections. This can be a space-separated or comma-separated list of IPv4 or IPv6 addresses. Alternatively, it can contain an entry of "*", in which case the REST API listens on all IP addresses.

The addresses that are bound to are listed in the error log. Addresses to which the REST API cannot be bound are also logged. If no addresses can be bound, the REST API shuts down.

Restricting Access to Trusted Users

In addition to username/password access, the *Restricting Access* section of the **System > Security** page provides the ability to further restrict access to the administrative capabilities of your Traffic Manager

system to a set of trusted IP addresses, CIDR subnets, or DNS wildcards. Access to the REST API is also affected by this capability.

Log Messages in Stingray

The Event Log

A number of specific API-related messages might be found in the Stingray event log under certain conditions:

- `REST API port changed: https://<URI>`

  Raised when the REST Daemon has been asked to change the port it listens on.

- `REST API started: https://<URI>`

  Raised when the REST Daemon starts.

- `REST API is shutting down`

  Raised when the REST Daemon closes down.

- `On IPv6 host but cannot set unspecified ip address to ::`

  Raised when the REST Daemon can't set itself up to listen on the IPv6 wildcard address.

- `Could not open traffic manager PID file for read: <error>`

  Raised when REST Daemon can't identify the Traffic Manager PID, and so can't signal it to reload its config after a change has been made via the REST API.

- `Could not open traffic manager PID file: <error>`

  Raised when REST Daemon can't identify the Traffic Manager PID, and so can't signal it to reload its config after a change has been made via the REST API.

- `Failed to write to audit log: <error>`

  Raised when the REST Daemon can't add lines to the audit log.

The Audit Log

The audit log records login attempts, configuration changes, and user logouts. It also records changes made using the Stingray Control API, and via the Traffic Manager CLI. Configuration changes made through the REST API follow the same behavior.

In addition to the typical configuration messages entered into the audit log, Stingray also provides the ability to track user activity in the REST API. It does this by grouping REST request/response exchanges made in close succession from a given user into a "session".

Stingray logs the first request in a group of one or more requests from a particular user/ip address combination in the audit log as a "session start". Requests received after the initial request are deemed to be part of the same user session. Then, after a specified timeout interval since the most recent request was received from the same user, a "session end" is logged.

CHAPTER 3 Examples and Use-Cases

Typical Usage

The following code samples demonstrate how to interact with the REST API for a variety of purposes. The examples are based on Perl using the `REST::Client` module to handle the connections to the Traffic Manager REST daemon.

Note: Further information on `REST::Client` can be found at the CPAN website: www.cpan.org

A typical Perl client connection might resemble the following:

```
#!/usr/bin/perl

use REST::Client;
use strict;

# Set up the connection
my $client = REST::Client->new();
$client->setHost( 'https://stingrayhost:9070' );
$client->addHeader( 'Authorization', 'Basic YWRtaW46am9iYmll' );
$client->addHeader( 'Content-Type', 'application/json' );

# Perform a HTTP GET on this URI
$client->GET( '/api/tm/2.0/config/active' );

# Print out the response body
print $client->responseContent();
```

In the above example, a new connection is established to the REST service on the Traffic Manager "`stingrayhost`" on port 9070.

The setHost() function allows us to set up a definitive hostname/port to which all requests are made. This is an optional feature, and the full hostname can be supplied when making the actual request if multiple hosts are required.

Two HTTP headers can be added here, one to provide *Basic Auth* authentication and the other to provide a declaration of the Content Type when making PUT requests. In the majority of cases, the content type is "`application/json`", apart from transactions involving raw files where it is necessary to use "`application/octet-stream`".

A GET request is sent to the REST service with a target of the resource URI as the supplied argument. Typically, the above script outputs a JSON structure showing the Traffic Manager resource tree at the top level:

```
{
    "children": [{
        "name": "rules",
        "href": "/api/tm/2.0/config/active/rules/"
    }, {
```

```
        "name": "actions",
        "href": "/api/tm/2.0/config/active/actions/"
    },
    ...
    (truncated)
    ...
   {
        "name": "auth",
        "href": "/api/tm/2.0/config/active/auth/"
    }]
}
```

Note: Each of the following examples make use of a further Perl module "`JSON`" in order to encode and decode between the JSON string used by `REST::Client` and a native Perl structure. This is done to simplify the parsing algorithm within the script. Further information regarding the `JSON` module can be found on the CPAN website at: www.cpan.org.

Listing Running Virtual Servers

In this example, we collect data on stored virtual servers by querying the `vservers` resource and identifying which ones are enabled (running).

The code structure is as follows:

- Instantiate a new REST Client object;
- Specify the hostname/port of the REST service to which all requests are to be directed;
- Add required HTTP headers for authentication and content type;
- Send a GET request for the `vservers` resource in order to return a list of all Virtual Servers on the system;
- Check the response body, and decode from JSON into a Perl structure. This value is a hash ref;
- Identify the `children` hash key, and iterate through the array to which it points;
- Each array item contains a hash of `name` and `href` associative values;
- Using the `name` value, perform a new GET request to return the full configuration for this named virtual server resource;
- Again, using the decoded JSON response body, identify the Boolean value of the `enabled` key in the `basic` configuration section. If it is `true`, this virtual server is running, so print it's name to STDOUT.

Important: This script does not contain any error checking in order to best demonstrate the basic functionality. It is strongly recommended you incorporate return value checking and other validation mechanisms as appropriate.

```
#!/usr/bin/perl

use REST::Client;
use JSON;
use strict;

# Set up the connection
my $client = REST::Client->new();
$client->setHost( 'https://stingrayhost:9070' );
$client->addHeader( 'Authorization', 'Basic YWRtaW46am9iYmll' );
$client->addHeader( 'Content-Type', 'application/json' );

# Request a list of all virtual servers
client->GET( '/api/tm/2.0/config/active/vservers' );

# Decode response into a perl structure for easy parsing
my $response = decode_json( $client->responseContent() );

# Obtain a reference to the children array
my $vsArrayRef = $response->{children};

# For each VS, make a request for its configuration and
# check the Boolean value of the 'enabled' key
foreach my $vs ( @$vsArrayRef ) {
   my $vsName = $vs->{name};
   $client->GET( "/api/tm/2.0/config/active/vservers/$vsName" );
   my $vsConfig = decode_json( $client->responseContent() );
   if( $vsConfig->{properties}->{basic}->{enabled} eq "true" ) {
      # Print the name of this matched VS
      print "$vsn\n";
   }
}
```

The expected output of a script such as this would be:

```
$ ./listVS.pl
Main Website
Intranet
Support Site
```

Adding a Node to a Pool

Provisioning systems can dynamically deploy applications across servers, perhaps in reaction to increased server load. This example demonstrates an application that modifies the nodes that a pool balances traffic to.

The code structure is as follows:

- Instantiate a new REST Client object;
- Specify the hostname/port of the REST service to which all requests are to be directed;
- Add required HTTP headers for authentication and content type;
- Send a GET request for the pool that the new node is added to. Check the response body, and decode from JSON into a Perl structure. This value is a hash ref;
- The new node must be added to the list of existing nodes before writing the data back to the pool resource. Failing to do this results in the existing array being overwritten with a single entry containing the new node name;
- Re-encode the perl structure into JSON and pass as an argument to the PUT request (using the pool name URI as the target);
- In this example, the script performs a check on the response code to ensure any problems are reported back (where the response code is not 200 OK);
- There is an optional portion of code at the end to iterate through the stored node list to ensure the new node name appears.

```
#!/usr/bin/perl -w

use REST::Client;
use JSON;
use strict;

# Set up the connection
my $client = REST::Client->new();
$client->setHost( 'http://localhost:9070' );
$client->addHeader( 'Authorization', 'Basic YWRtaW46am9iYmll' );
$client->addHeader( 'Content-Type', 'application/json' );

# Our pool and new node details
my $poolName = "WebPool";
my $newNode = "www3.riverbed.com:80";

# Get the config for the pool in question
$client->GET( "/api/tm/2.0/config/active/pools/$poolName" );
my $poolConfig = decode_json( $client->responseContent() );

# Find the existing nodes list (a hashref), and add our new node
my $nodesRef = $poolConfig->{properties}->{basic}->{nodes};
push @$nodesRef, $newNode;
```

```
# Re-encode as a JSON string
my $poolStr = encode_json( $poolConfig );

# Now send a PUT request to the REST service
$client->PUT( "/api/tm/2.0/config/active/pools/$poolName",
              $poolStr );

# Print out the response code if we were NOT successful
if( $client->responseCode() ne '200' ) {
   die "FAILED with HTTP code: " . $client->responseCode();
}

# We're done! Verify that the node has been added
$client->GET( "/api/tm/2.0/config/active/pools/$poolName" );
$poolConfig = decode_json( $client->responseContent() );
print "Stored nodes for pool '$poolName':\n";
foreach my $node ( @{$poolConfig->{properties}->{basic}->{nodes}} )
{
   print "$node\n";
}
```

The expected output of a script such as this would be:

```
$ ./addNode.pl
Stored nodes for pool 'WebPool':
www1.riverbed.com:80
www2.riverbed.com:80
www3.riverbed.com:80
```

CHAPTER 4 Resource Model Reference

About the Resource Model Reference

This chapter lists all the configuration, counter, and information resources available through the REST API model.

Each section relates to a specific resource and lists its name, description, and unique URI path, and provides a table of properties.

For each property, you can find the description and data type. Additional information is provided where applicable, such as default value, permitted values (for enumerated types), and SNMP counter name. For Table-type properties, a list of the Primary and Sub keys is provided.

The path to use in your URIs is listed for each resource. For example, the *URI path* for a virtual server configuration resource is `virtual_servers`, so to address a stored virtual server named "foo", you would use:

```
/api/tm/2.0/config/active/virtual_servers/foo
```

Configuration Resources

Action Program

URI Path: `action_programs`

This is a program or script that can be referenced and used by actions of type 'Program'

Property	Description
There are no properties to display for this resource.	

Alerting Action

URI Path: `actions`

A response to an event occurring in your traffic manager. An example of an action might be sending an email or writing a line to a log file.

Property	Description
`note`	A description of the action.

<table>
<tr><td></td><td>Value type: <code>FreeformString</code>

Default: <code><none></code></td></tr>
<tr><td><code>syslog_msg_len_limit</code></td><td>Maximum length in bytes of a message sent to the remote syslog. Messages longer than this will be truncated before they are sent.

Value type: <code>UInt</code>

Default: <code>1024</code></td></tr>
<tr><td><code>timeout</code></td><td>How long the action can run for before it is stopped automatically (set to 0 to disable timeouts).

Value type: <code>UInt</code>

Default: <code>60</code></td></tr>
<tr><td><code>type</code></td><td>The action type.

Value type: <code>Enum(String)</code>

Default: <code><none></code>

Permitted values:
<table>
<tr><td><code>email</code></td><td>E-Mail</td></tr>
<tr><td><code>log</code></td><td>Log to File</td></tr>
<tr><td><code>program</code></td><td>Program</td></tr>
<tr><td><code>soap</code></td><td>SOAP Callback</td></tr>
<tr><td><code>syslog</code></td><td>Log to Syslog</td></tr>
<tr><td><code>trap</code></td><td>SNMP Notify or Trap</td></tr>
</table></td></tr>
<tr><td><code>verbose</code></td><td>Enable or disable verbose logging for this action.

Value type: <code>Boolean</code>

Default: <code>false</code></td></tr>
<tr><td colspan="2">Properties for the <code>email</code> section:</td></tr>
<tr><td><code>server</code></td><td>The SMTP server to which messages should be sent. This must be a valid IPv4 address or resolvable hostname (with optional port).

Value type: <code>String</code>

Default: <code><none></code></td></tr>
</table>

<table>
<tr><td>to</td><td>A set of e-mail addresses to which messages will be sent.
Value type: Set(String)
Default: <none></td></tr>
<tr><td colspan="2">Properties for the log section:</td></tr>
<tr><td>file</td><td>The full path of the file to log to. The text %zeushome% will be replaced with the location where the software is installed.
Value type: String
Default: <none></td></tr>
<tr><td>from</td><td>The e-mail address from which messages will appear to originate.
Value type: String
Default: stingraytrafficmanager@%hostname%</td></tr>
<tr><td colspan="2">Properties for the program section:</td></tr>
<tr><td>arguments</td><td>A table containing arguments and argument values to be passed to the event handling program.
primary key:<table><tr><td>name (String)</td><td>The name of the argument to be passed to the event handling program.</td></tr></table>sub keys:<table><tr><td>value (String)</td><td>The value of the argument to be passed to the event handling program.</td></tr><tr><td>description (String)</td><td>A description for the argument provided to the program.</td></tr></table></td></tr>
<tr><td>program</td><td>The program to run.
Value type: String
Default: <none></td></tr>
<tr><td colspan="2">Properties for the soap section:</td></tr>
<tr><td>additional_data</td><td>Additional information to send with the SOAP call.
Value type: String</td></tr>
</table>

	Default: <none>
password	The password for HTTP basic authentication. Value type: Password Default: <none>
proxy	The address of the server implementing the SOAP interface (For example, https://example.com). Value type: String Default: <none>
username	Username for HTTP basic authentication. Leave blank if you do not wish to use authentication. Value type: String Default: <none>
Properties for the syslog section:	
sysloghost	The host and optional port to send syslog messages to (if empty, messages will be sent to localhost). Value type: String Default: <none>
Properties for the trap section:	
auth_password	The authentication password for sending a Notify over SNMPv3. Blank to send unauthenticated traps. Value type: Password Default: <none>
community	The community string to use when sending a Trap over SNMPv1 or a Notify over SNMPv2c. Value type: String Default: <none>
hash_algorithm	The hash algorithm for SNMPv3 authentication.

<table>
<tr><td></td><td>Value type: <code>Enum(String)</code>

Default: <code>md5</code>

Permitted values:
<table>
<tr><td><code>md5</code></td><td>MD5</td></tr>
<tr><td><code>sha1</code></td><td>SHA-1</td></tr>
</table>
</td></tr>
<tr><td><code>priv_password</code></td><td>The encryption password to encrypt a Notify message for SNMPv3. Requires that authentication also be configured. Blank to send unencrypted traps.

Value type: <code>Password</code>

Default: <code><none></code></td></tr>
<tr><td><code>traphost</code></td><td>The hostname or IPv4 address and optional port number that should receive traps.

Value type: <code>String</code>

Default: <code><none></code></td></tr>
<tr><td><code>username</code></td><td>The SNMP username to use to send the Notify over SNMPv3.

Value type: <code>String</code>

Default: <code><none></code></td></tr>
<tr><td><code>version</code></td><td>The SNMP version to use to send the Trap/Notify.

Value type: <code>Enum(String)</code>

Default: <code>snmpv1</code>

Permitted values:
<table>
<tr><td><code>snmpv1</code></td><td>SNMPv1</td></tr>
<tr><td><code>snmpv2c</code></td><td>SNMPv2c</td></tr>
<tr><td><code>snmpv3</code></td><td>SNMPv3</td></tr>
</table>
</td></tr>
</table>

Aptimizer Application Scope

URI Path: `aptimizer/scopes`

Application scopes define criteria that match URLs to specific logical web applications hosted by a virtual server.

Property	Description
`hostnames`	The hostnames to limit acceleration to. Value type: `Set(String)` Default: `<none>`
`root`	The root path of the application defined by this application scope. Value type: `String` Default: `/`

Aptimizer Profile

URI Path: `aptimizer/profiles`

An Aptimizer profile can be applied to a HTTP virtual server to enable automatic web content optimization.

Property	Description
`background_after`	If Aptimizer can finish optimizing the resource within this time limit then serve the optimized content to the client, otherwise complete the optimization in the background and return the original content to the client. If set to 0, Aptimizer will always wait for the optimization to complete before sending a response to the client. Value type: `UInt` Default: `<none>`

<table>
<tr><td>background_on_additional_resources</td><td>If a web page contains resources that have not yet been optimized, fetch and optimize those resources in the background and send the original web page to clients until the optimized page and resources are ready.

Value type: Boolean

Default: false</td></tr>
<tr><td>config</td><td>Placeholder to be overwritten when we have Aptimizer support in RESTD

Value type: String

Default: {}</td></tr>
<tr><td>mode</td><td>Set the Aptimizer mode to turn acceleration on or off.

Value type: Enum(String)

Default: active

Permitted values:<table><tr><td>active</td><td>On - Aptimizer acceleration is enabled</td></tr><tr><td>idle</td><td>Off - Acceleration is disabled, but requests for Aptimizer resources are served</td></tr><tr><td>stealth</td><td>Stealth - Acceleration is controlled by a cookie</td></tr></table></td></tr>
<tr><td>show_info_bar</td><td>Show the Aptimizer information bar on aptimized web pages.

Value type: Boolean

Default: false</td></tr>
</table>

Bandwidth Class

URI Path: `bandwidth`

A Bandwidth class, which can be assigned to a virtual server or pool in order to limit the number of bytes per second used by inbound or outbound traffic.

Property	Description
`maximum`	The maximum bandwidth to allocate to connections that are associated with this bandwidth class (in kbits/second). Value type: `UInt` Default: `10000`
`note`	A description of this bandwidth class. Value type: `FreeformString` Default: `<none>`
`sharing`	The scope of the bandwidth class. Value type: `Enum(String)` Default: `cluster` Permitted values: `cluster` — Bandwidth is shared across all traffic managers `connection` — Each connection can use the maximum rate `machine` — Bandwidth is shared per traffic manager

Cloud Credentials

URI Path: `cloud_api_credentials`

Cloud credentials used in cloud API calls

Property	Description
`api_server`	The vCenter server hostname or IP address.

	Value type: `String` Default: `<none>`
`cloud_api_timeout`	The traffic manager creates and destroys nodes via API calls. This setting specifies (in seconds) how long to wait for such calls to complete. Value type: `UInt` Default: `200`
`cred1`	The first part of the credentials for the cloud user. Typically this is some variation on the username concept. Value type: `String` Default: `<none>`
`cred2`	The second part of the credentials for the cloud user. Typically this is some variation on the password concept. Value type: `Password` Default: `<none>`
`cred3`	The third part of the credentials for the cloud user. Typically this is some variation on the authentication token concept. Value type: `Password` Default: `<none>`
`script`	The script to call for communication with the cloud API. Value type: `String` Default: `<none>`
`update_interval`	The traffic manager will periodically check the status of the cloud through an API call. This setting specifies the interval between such updates. Value type: `UInt` Default: `30`

Custom configuration set

URI Path: `custom`

Custom configuration sets store arbitrary named values. These values can be read by SOAP or REST clients.

Property	Description
`string_lists`	This table contains named lists of strings primary key: `name (String)` — Name of list sub keys: `value (List(String))` — Named list of user-specified strings.

Event Type

URI Path: `event_types`

Configuration that ties actions to a set of events that trigger them.

Property	Description
`actions`	The actions triggered by events matching this event type, as a list of action references. Value type: `List(Reference(config-event-action))` Default: `<none>`
`built_in`	If set to `Yes` this indicates that this configuration is built-in (provided as part of the software) and must not be deleted or edited. Value type: `Boolean` Default: `false`
`note`	A description of this event type. Value type: `FreeformString` Default: `<none>`

Properties for the `cloudcredentials` section:	
`event_tags`	Cloud credentials event tags Value type: `List(String)` Default: `<none>`
`objects`	Cloud credentials object names Value type: `List(String)` Default: `<none>`
Properties for the `config` section:	
`event_tags`	Configuration file event tags Value type: `List(String)` Default: `<none>`
Properties for the `faulttolerance` section:	
`event_tags`	Fault tolerance event tags Value type: `List(String)` Default: `<none>`
Properties for the `general` section:	
`event_tags`	General event tags Value type: `List(String)` Default: `<none>`
Properties for the `glb` section:	
`event_tags`	GLB service event tags Value type: `List(String)` Default: `<none>`
`objects`	GLB service object names

	Value type: `List(String)` Default: `<none>`
Properties for the `java` section:	
`event_tags`	Java event tags Value type: `List(String)` Default: `<none>`
Properties for the `licensekeys` section:	
`event_tags`	License key event tags Value type: `List(String)` Default: `<none>`
`objects`	License key object names Value type: `List(String)` Default: `<none>`
Properties for the `locations` section:	
`event_tags`	Location event tags Value type: `List(String)` Default: `<none>`
`objects`	Location object names Value type: `List(String)` Default: `<none>`
Properties for the `monitors` section:	
`event_tags`	Monitor event tags Value type: `List(String)` Default: `<none>`

`objects`	Monitors object names Value type: `List(String)` Default: `<none>`
Properties for the `pools` section:	
`event_tags`	Pool key event tags Value type: `List(String)` Default: `<none>`
`objects`	Pool object names Value type: `List(String)` Default: `<none>`
Properties for the `protection` section:	
`event_tags`	Service protection class event tags Value type: `List(String)` Default: `<none>`
`objects`	Service protection class object names Value type: `List(String)` Default: `<none>`
Properties for the `rules` section:	
`event_tags`	Rule event tags Value type: `List(String)` Default: `<none>`
`objects`	Rule object names Value type: `List(String)` Default: `<none>`

Properties for the `slm` section:	
`event_tags`	SLM class event tags Value type: `List(String)` Default: `<none>`
`objects`	SLM class object names Value type: `List(String)` Default: `<none>`
Properties for the `ssl` section:	
`event_tags`	SSL event tags Value type: `List(String)` Default: `<none>`
Properties for the `sslhw` section:	
`event_tags`	SSL hardware event tags Value type: `List(String)` Default: `<none>`
Properties for the `trafficscript` section:	
`event_tags`	TrafficScript event tags Value type: `List(String)` Default: `<none>`
Properties for the `vservers` section:	
`event_tags`	Virtual server event tags Value type: `List(String)` Default: `<none>`
`objects`	Virtual server object names

	Value type: `List(String)` Default: `<none>`
Properties for the `zxtms` section:	
`event_tags`	Traffic manager event tags Value type: `List(String)` Default: `<none>`
`objects`	Traffic manager object names Value type: `List(String)` Default: `<none>`

Extra File

URI Path: `extra_files`

A user-uploaded file. Such files can be used in TrafficScript code using the `resource.get` function.

Property	Description
There are no properties to display for this resource.	

GLB Service

URI Path: `glb_services`

A global load balancing service is used by a virtual server to modify DNS requests in order load balance data across different GLB locations.

Property	Description
`algorithm`	Defines the global load balancing algorithm to be used. Value type: `Enum(String)` Default: `hybrid`

<table>
<tr><td></td><td>Permitted values:<table>
<tr><td><code>chained</code></td><td>Sends traffic to one location at a time, until that location fails where the next one in the chain is used.</td></tr>
<tr><td><code>geo</code></td><td>Distributes traffic based solely on the geographic location of each client.</td></tr>
<tr><td><code>hybrid</code></td><td>Distribute traffic based on both the load and geographic location.</td></tr>
<tr><td><code>load</code></td><td>Distributes traffic based on the current load to each location.</td></tr>
<tr><td><code>round_robin</code></td><td>Distributes traffic by assigning each request to a new location in turn. Over a period of time, all locations will receive the same number of requests.</td></tr>
<tr><td><code>weighted_random</code></td><td>Distributes traffic in a random way, but according to a weighted policy defined by individual location weights</td></tr>
</table></td></tr>
<tr><td><code>all_monitors_needed</code></td><td>Are all the monitors required to be working in a location to mark this service as alive?

Value type: <code>Boolean</code>

Default: <code>true</code></td></tr>
<tr><td><code>chained_auto_failback</code></td><td>Enable/Disable automatic failback mode.

Value type: <code>Boolean</code>

Default: <code>false</code></td></tr>
<tr><td><code>chained_location_order</code></td><td>The locations this service operates for and defines the order in which locations fail.

Value type: <code>List(String)</code>

Default: <code><none></code></td></tr>
</table>

<table>
<tr><td>dnssec_keys</td><td>A table mapping domains to the private keys that authenticate them

primary key:<table><tr><td>domain (String)</td><td>A domain authenticated by the associated private keys.</td></tr></table>
sub keys:<table><tr><td>ssl_key (Set(String))</td><td>Private keys that authenticate the associated domain.</td></tr></table></td></tr>
<tr><td>domains</td><td>The domains shown here should be a list of Fully Qualified Domain Names that you would like to balance globally. Responses from the back end DNS servers for queries that do not match this list will be forwarded to the client unmodified. Note: "*" may be used as a wild card.

Value type: Set(String)

Default: <none></td></tr>
<tr><td>enabled</td><td>Enable/Disable our response manipulation of DNS.

Value type: Boolean

Default: false</td></tr>
<tr><td>geo_effect</td><td>How much should the locality of visitors affect the choice of location used? This value is a percentage, 0% means that no locality information will be used, and 100% means that locality will always control which location is used. Values between the two extremes will act accordingly.

Value type: UInt

Default: 50</td></tr>
<tr><td>location_draining</td><td>This is the list of locations for which this service is draining. A location that is draining will never serve any of its local IPs for this domain. This can be used to take a location off-line.

Value type: Set(String)

Default: <none></td></tr>
<tr><td>location_settings</td><td>Table containing location specific settings.

primary key:<table><tr><td>location (String)</td><td>Location to which the associated settings apply.</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td>sub keys:<table>
<tr><td>weight (UInt)</td><td>Weight for this location, for use by the weighted random algorithm.</td></tr>
<tr><td>ips (Set(String))</td><td>The IP addresses that are present in a location. If the Global Load Balancer decides to direct a DNS query to this location, then it will filter out all IPs that are not in this list.</td></tr>
<tr><td>monitors (Set(String))</td><td>The monitors that are present in a location.</td></tr>
</table></td></tr>
<tr><td>return_ips_on_fail</td><td>Return all or none of the IPs under complete failure.
Value type: Boolean
Default: true</td></tr>
<tr><td>rules</td><td>Response rules to be applied in the context of the service, in order, comma separated.
Value type: List(Reference(config-trafficscript))
Default: <none></td></tr>
<tr><td>ttl</td><td>The TTL that should be used for the domains handled by this config, or -1 if the original TTL should be left as is.
Value type: Int
Default: -1</td></tr>
<tr><td colspan="2">Properties for the log section:</td></tr>
<tr><td>enabled</td><td>Log connections to this GLB service?
Value type: Boolean
Default: false</td></tr>
<tr><td>filename</td><td>The filename the verbose query information should be logged to. Appliances will ignore this.
Value type: String
Default: %zeushome%/zxtm/log/services/%g.log</td></tr>
</table>

format	The format of the log lines. Value type: String Default: %s %l %q %g %n %d %a

Global Settings

URI Path: global_settings

General settings that apply to every machine in the cluster.

Property	Description
accepting_delay	How often, in milliseconds, each traffic manager child process (that isn't listening for new connections) checks to see whether it should start listening for new connections. Value type: UInt Default: 50
afm_enabled	Is the Application Firewall enabled. Value type: Boolean Default: false
child_control_command_timeout	Timeout for waiting for child processes to respond to parent control requests If a child process (zeus.zxtm, zeus.eventd, zeus.autoscaler, etc) takes longer than this number of seconds to respond to a parent control command, an error message will be logged in the event log. Value type: UInt Default: 10
child_control_kill_timeout	Timeout for waiting for child processes to respond to parent control requests If a child process (zeus.zxtm, zeus.eventd, zeus.autoscaler, etc) takes longer than this number of seconds to respond to a parent control command, then the parent zeus.zxtm process will assume this process is stuck in an unresponsive loop and will kill it, log the termination event, and wait for a new process of the same type to restart. Set this to 0 to disable killing unresponsive child processes.

	Value type: `UInt` Default: `60`
`chunk_size`	The default chunk size for reading/writing requests. Value type: `UInt` Default: `4096`
`client_first_opt`	Whether or not your traffic manager should make use of TCP optimisations to defer the processing of new client-first connections until the client has sent some data. Value type: `Boolean` Default: `false`
`max_fds`	The maximum number of file descriptors that your traffic manager will allocate. Value type: `UInt` Default: `1048576`
`monitor_memory_size`	The maximum number of nodes that can be monitored. This is used to size the shared memory, that keeps track of the state. Value type: `UInt` Default: `4096`
`rate_class_limit`	The maximum number of Rate classes that can be created. Approximately 100 bytes will be pre-allocated per Rate class. Value type: `UInt` Default: `25000`
`shared_pool_size`	The size of the shared memory pool used for shared storage across worker processes (e.g. bandwidth shared data).This is specified as either a percentage of system RAM, `5%` for example, or an absolute size such as `10MB`. Value type: `String` Default: `10MB`

<table>
<tr><td>slm_class_limit</td><td>The maximum number of SLM classes that can be created. Approximately 100 bytes will be pre-allocated per SLM class.

Value type: UInt

Default: 1024</td></tr>
<tr><td>so_rbuff_size</td><td>The size of the operating system's read buffer. A value of 0 (zero) means to use the OS default; in normal circumstances this is what should be used.

Value type: UInt

Default: <none></td></tr>
<tr><td>so_wbuff_size</td><td>The size of the operating system's write buffer. A value of 0 (zero) means to use the OS default; in normal circumstances this is what should be used.

Value type: UInt

Default: <none></td></tr>
<tr><td>socket_optimizations</td><td>Whether or not the traffic manager should use potential network socket optimisations. If set to auto, a decision will be made based on the host platform.

Value type: Enum(String)

Default: auto

Permitted values:<table><tr><td>auto</td><td>Decide based on local platform</td></tr><tr><td>no</td><td>Disable socket optimizations</td></tr><tr><td>yes</td><td>Enable socket optimizations</td></tr></table></td></tr>
<tr><td>storage_shared</td><td>Whether the storage for the traffic managers' configuration is shared between cluster members.

Value type: Boolean

Default: false</td></tr>
<tr><td>tip_class_limit</td><td>The maximum number of Traffic IP Groups that can be created.

Value type: UInt

Default: 10000</td></tr>
</table>

<table>
<tr><th colspan="2">Properties for the <code>admin</code> section:</th></tr>
<tr><td><code>ssl3_allow_rehandshake</code></td><td>Whether or not SSL3/TLS re-handshakes should be supported for admin server and internal connections.

Value type: <code>Enum(String)</code>

Default: <code>rfc5746</code>

Permitted values:
<table>
<tr><td><code>always</code></td><td>Always allow</td></tr>
<tr><td><code>never</code></td><td>Never allow</td></tr>
<tr><td><code>rfc5746</code></td><td>Only if client uses RFC 5746 (Secure Renegotiation Extension)</td></tr>
<tr><td><code>safe</code></td><td>Allow safe re-handshakes</td></tr>
</table></td></tr>
<tr><td><code>ssl3_ciphers</code></td><td>The SSL ciphers to use for admin server and internal connections. For information on supported ciphers see the online help.

Value type: <code>String</code>

Default: <code>SSL_RSA_WITH_AES_128_CBC_SHA,SSL_RSA_WITH_AES_256_CBC_SHA,SSL_RSA_WITH_3DES_EDE_CBC_SHA</code></td></tr>
<tr><td><code>ssl3_diffie_hellman_key_length</code></td><td>The length in bits of the Diffie-Hellman key for ciphers that use Diffie-Hellman key agreement for admin server and internal connections.

Value type: <code>Enum(UInt)</code>

Default: <code>dh_2048</code>

Permitted values:
<table>
<tr><td><code>dh_1024</code></td><td>Use 1024 bit keys for Diffie-Hellman ciphers.</td></tr>
<tr><td><code>dh_2048</code></td><td>Use 2048 bit keys for Diffie-Hellman ciphers.</td></tr>
<tr><td><code>dh_3072</code></td><td>Use 3072 bit keys for Diffie-Hellman ciphers.</td></tr>
<tr><td><code>dh_4096</code></td><td>Use 4096 bit keys for Diffie-Hellman ciphers.</td></tr>
</table></td></tr>
<tr><td><code>ssl3_min_rehandshake</code></td><td>If SSL3/TLS re-handshakes are supported on the admin server, this</td></tr>
</table>

_interval	defines the minimum time interval (in milliseconds) between handshakes on a single SSL3/TLS connection that is permitted. To disable the minimum interval for handshakes the key should be set to the value 0. Value type: UInt Default: 1000
ssl_insert_extra_fragment	Whether or not SSL3 and TLS1 use one-byte fragments as a BEAST countermeasure for admin server and internal connections. Value type: Boolean Default: false
ssl_max_handshake_message_size	The maximum size (in bytes) of SSL handshake messages that the admin server and internal connections will accept. To accept any size of handshake message the key should be set to the value 0. Value type: UInt Default: 10240
ssl_prevent_timing_side_channels	Take performance degrading steps to prevent exposing timing side-channels with SSL3 and TLS used by the admin server and internal connections. Value type: Boolean Default: false
support_ssl2	Whether or not SSL2 support is enabled for admin server and internal connections. Value type: Boolean Default: false
support_ssl3	Whether or not SSL3 support is enabled for admin server and internal connections. Value type: Boolean Default: false
support_tls1	Whether or not TLS1.0 support is enabled for admin server and internal connections.

	Value type: `Boolean` Default: `true`
`support_tls11`	Whether or not TLS1.1 support is enabled for admin server and internal connections. Value type: `Boolean` Default: `true`
Properties for the `appliance` section:	
`bootloader_password`	The password used to protect the bootloader. An empty string means there will be no protection. Value type: `Password` Default: `<none>`
`manage_ncipher`	Whether or not we should manage the nCipher Support Software automatically. Value type: `Boolean` Default: `true`
`nethsm_esn`	The ESN (electronic serial number) for the NetHSM. Value type: `String` Default: `<none>`
`nethsm_hash`	The key hash for the NetHSM. Value type: `String` Default: `<none>`
`nethsm_ip`	The IP address of the nCipher NetHSM to use. Value type: `String` Default: `<none>`
`nethsm_ncipher_rfs`	The IP address of the nCipher Remote File System to use. Value type: `String`

	Default: `<none>`
`return_path_routing_enabled`	Whether or not the traffic manager will attempt to route response packets back to clients via the same route on which the corresponding request arrived. Note that this applies only to the last hop of the route - the behaviour of upstream routers cannot be altered by the traffic manager. Value type: `Boolean` Default: `false`
Properties for the `aptimizer` section:	
`cache_entry_lifetime`	The period of time (in seconds) that unaccessed cache entries will be retained by aptimizer. Value type: `UInt` Default: `86400`
`cache_entry_limit`	The maximum number of cache entries that will be retained by aptimizer before removing old entries to make room for new ones. Value type: `UInt` Default: `20000`
`default_profile`	The Profile to use by default if no mappings are configured (or if Aptimizer is licensed in Express mode) Value type: `String` Default: `Express`
`default_scope`	The Scope to use by default if no mappings are configured (or if Aptimizer is licensed in Express mode) Value type: `String` Default: `Any hostname or path`
`dependent_fetch_timeout`	How long to wait for dependent resource fetches (default 30 seconds). Value type: `UInt` Default: `30`

enable_state_dump	Whether or not the Aptimizer state will be dumped if "/aptimizer-state-dump" is appended to an Aptimized URL. Value type: Boolean Default: false
ipc_timeout	The time after which connections between the traffic manager and Aptimizer processes will be closed, should an optimization job take considerably longer than expected. Value type: UInt Default: 120
max_original_content_buffer_size	The maximum size of unoptimized content buffered in the traffic manager for a single backend response that is undergoing Aptimizer optimization. Responses larger than this will not be optimized. Note that if the backend response is compressed then this setting pertains to the compressed size, before Aptimizer decompresses it. Units of KB and MB can be used, no postfix denotes bytes. Value range is 1 - 128MB. Value type: String Default: 2MB
queue_buffer_size	The size in bytes of the operating system buffer which is used to send request URLs and data to Aptimizer and return optimized resources from Aptimizer. A larger buffer will allow a greater number of simultaneous resources to be optimized, particularly if a large number of requests are made at the same time, for example an HTML page containing hundreds of images to optimize. If this is set to zero, the default operating system buffer size will be used. Value type: UInt Default: 131072
resource_lifetime	The period of time (in seconds) that resource data is retained by aptimizer after it is no longer actively in use. Value type: UInt Default: 10
resource_memory_limit	The maximum amount of memory the cache is allowed to have pinned. Once it goes over that limit, it starts releasing resource data in LRU order.

	Value type: `UInt` Default: `256`
`watchdog_interval`	The period of time (in seconds) after which a previous failure will no longer count towards the watchdog limit. Value type: `UInt` Default: `300`
`watchdog_limit`	The maximum number of times the Aptimizer sub-process will be started or restarted within the interval defined by the aptimizer!watchdog_interval setting. If the process fails this many times, it must be restarted manually from the Diagnose page. Zero means no limit. Value type: `UInt` Default: `3`
Properties for the `bandwidth` section:	
`license_sharing`	For the global BW limits, how the bandwidth allocation should be shared between consumers. In 'pooled' mode, the allocation is shared between all consumers, who can write as much data as they want until the pool of data is exhausted. In 'quota' mode, bandwidth is divided between consumers, who can write only as much as they are allocated. Any unused bandwidth will be lost. Value type: `Enum(String)` Default: `pooled` Permitted values: `pooled` (pooled); `quota` (quota)
`pooled_min_write`	For the global BW limits using 'pooled' bandwidth allocation sharing between consumers, when the license limit is reached the allowance will be evenly distributed between the remaining consumers. Each consumer will, however be permitted to write at least this much data. Value type: `UInt` Default: `4096`
Properties for the `cluster_comms` section:	

allow_update_default	The default value of allow_update for new cluster members. If you have cluster members joining from less trusted locations (such as cloud instances) this can be set to false in order to make them effectively "read-only" cluster members. Value type: Boolean Default: true
allowed_update_hosts	The hosts that can contact the internal administration port on each traffic manager. This should be a list containing IP addresses, CIDR IP subnets, and localhost; or it can be set to all to allow any host to connect. Value type: List(String) Default: all
state_sync_interval	How often to propagate the session persistence and bandwidth information to other traffic managers in the same cluster. Set this to 0 (zero) to disable propagation. Note that a cluster using "unicast" heartbeat messages cannot turn off these messages. Value type: UInt Default: 3
state_sync_timeout	The maximum amount of time to wait when propagating session persistence and bandwidth information to other traffic managers in the same cluster. Once this timeout is hit the transfer is aborted and a new connection created. Value type: UInt Default: 6
Properties for the connection section:	
idle_connections_max	The maximum number of unused HTTP keepalive connections with back-end nodes that the traffic manager should maintain for re-use. Setting this to 0 (zero) will cause the traffic manager to auto-size this parameter based on the available number of file-descriptors. Value type: UInt Default: <none>

idle_timeout	How long an unused HTTP keepalive connection should be kept before it is discarded. Value type: UInt Default: 10
listen_queue_size	The listen queue size for managing incoming connections. It may be necessary to increase the system's listen queue size if this value is altered. If the value is set to 0 then the default system setting will be used. Value type: UInt Default: <none>
max_accepting	Number of processes that should accept new connections. Only this many traffic manager child processes will listen for new connections at any one time. Setting this to 0 (zero) will cause your traffic manager to select an appropriate default value based on the architecture and number of CPUs. Value type: UInt Default: <none>
multiple_accept	Whether or not the traffic manager should try to read multiple new connections each time a new client connects. This can improve performance under some very specific conditions. However, in general it is recommended that this be set to 'false'. Value type: Boolean Default: false
Properties for the dns section:	
max_ttl	Maximum Time To Live (expiry time) for entries in the DNS cache. Value type: UInt Default: 86400
min_ttl	Minimum Time To Live (expiry time) for entries in the DNS cache. Value type: UInt Default: 86400

negative_expiry	Expiry time for failed lookups in the DNS cache. Value type: UInt Default: 60
size	Maximum number of entries in the DNS cache. Value type: UInt Default: 10867
timeout	Timeout for receiving a response from a DNS server. Value type: UInt Default: 12
Properties for the ec2 section:	
access_key_id	Amazon EC2 Access Key ID. Value type: String Default: <none>
secret_access_key	Amazon EC2 Secret Access Key. Value type: Password Default: <none>
vpc_decluster_on_stop	Whether to decluster the traffic manager running inside vpc when the instance stops. Value type: Boolean Default: false
Properties for the eventing section:	
mail_interval	The minimum length of time that must elapse between alert emails being sent. Where multiple alerts occur inside this timeframe, they will be retained and sent within a single email rather than separately. Value type: UInt Default: 30

`max_attempts`	The number of times to attempt to send an alert email before giving up. Value type: `UInt` Default: `10`
Properties for the `fault_tolerance` section:	
`arp_count`	The number of ARP packets a traffic manager should send when an IP address is raised. Value type: `UInt` Default: `10`
`auto_failback`	Whether or not traffic IPs automatically move back to machines that have recovered from a failure and have dropped their traffic IPs. Value type: `Boolean` Default: `true`
`frontend_check_ips`	The IP addresses used to check front-end connectivity. Set this to an empty string if the traffic manager is on an Intranet with no external connectivity. Value type: `Set(String)` Default: `%gateway%`
`heartbeat_method`	The method traffic managers should use to exchange cluster heartbeat messages. Value type: `Enum(String)` Default: `unicast` Permitted values: `multicast` (multicast); `unicast` (unicast)
`monitor_interval`	The frequency, in milliseconds, that each traffic manager machine should check and announce its connectivity. Value type: `UInt` Default: `500`

monitor_timeout	How long, in seconds, each traffic manager should wait for a response from its connectivity tests or from other traffic manager machines before registering a failure. Value type: UInt Default: 5
multicast_address	The multicast address and port to use to exchange cluster heartbeat messages. Value type: String Default: 239.100.1.1:9090
unicast_port	The unicast UDP port to use to exchange cluster heartbeat messages. Value type: UInt Default: 9090
use_bind_ip	Whether or not cluster heartbeat messages should only be sent and received over the management network. Value type: Boolean Default: false
verbose	Whether or not a traffic manager should log all connectivity tests. This is very verbose, and should only be used for diagnostic purposes. Value type: Boolean Default: false
Properties for the fips section:	
enabled	Enable FIPS Mode (requires software restart). Value type: Boolean Default: false
Properties for the ftp section:	
data_bind_low	Whether or not the traffic manager should permit use of FTP data connection source ports lower than 1024. If No the traffic manager can completely drop root privileges, if Yes some or all privileges may be

<table>
<tr><td></td><td>retained in order to bind to low ports.

Value type: <code>Boolean</code>

Default: <code>false</code></td></tr>
<tr><td colspan="2">Properties for the <code>glb</code> section:</td></tr>
<tr><td><code>verbose</code></td><td>Write a message to the logs for every DNS query that is load balanced, showing the source IP address and the chosen datacenter.

Value type: <code>Boolean</code>

Default: <code>false</code></td></tr>
<tr><td colspan="2">Properties for the <code>historical_activity</code> section:</td></tr>
<tr><td><code>keep_days</code></td><td>Number of days to store historical traffic information, if set to <code>0</code> the data will be kept indefinitely.

Value type: <code>UInt</code>

Default: <code>90</code></td></tr>
<tr><td colspan="2">Properties for the <code>ip</code> section:</td></tr>
<tr><td><code>appliance_returnpath</code></td><td>A table of MAC to IP address mappings for each router where return path routing is required.

primary key:
<table><tr><td><code>mac</code>
<code>(String)</code></td><td>The MAC address of a router the software is connected to.</td></tr></table>
sub keys:
<table><tr><td><code>ipv4</code>
<code>(String)</code></td><td>The MAC address to IPv4 address mapping of a router the software is connected to. The <code>*</code> (asterisk) in the key name is the MAC address, the value is the IP address.</td></tr><tr><td><code>ipv6</code>
<code>(String)</code></td><td>The MAC address to IPv6 address mapping of a router the software is connected to. The <code>*</code> (asterisk) in the key name is the MAC address, the value is the IP address.</td></tr></table></td></tr>
<tr><td colspan="2">Properties for the <code>java</code> section:</td></tr>
<tr><td><code>classpath</code></td><td>CLASSPATH to use when starting the Java runner.

Value type: <code>String</code></td></tr>
</table>

	Default: <none>
command	Java command to use when starting the Java runner, including any additional options. Value type: String Default: java -server
enabled	Whether or not Java support should be enabled. If this is set to No, then your traffic manager will not start any Java processes. Java support is only required if you are using the TrafficScript java.run() function. Value type: Boolean Default: true
lib	Java library directory for additional jar files. The Java runner will load classes from any .jar files stored in this directory, as well as the * jar files and classes stored in traffic manager's catalog. Value type: String Default: <none>
max_connections	Maximum number of simultaneous Java requests. If there are more than this many requests, then further requests will be queued until the earlier requests are completed. This setting is per-CPU, so if your traffic manager is running on a machine with 4 CPU cores, then each core can make this many requests at one time. Value type: UInt Default: 256
session_age	Default time to keep a Java session. Value type: UInt Default: 86400
Properties for the log section:	
error_level	The minimum severity of events/alerts that should be logged to disk. INFO will log all events; a higher severity setting will log fewer events. More fine-grained control can be achieved using events and actions.

<table>
<tr><td></td><td>Value type: <code>Enum(UInt)</code>

Default: <code>info</code>

Permitted values:
<table>
<tr><td><code>fatal</code></td><td>Only fatal errors are logged</td></tr>
<tr><td><code>info</code></td><td>All events are logged to disk</td></tr>
<tr><td><code>serious</code></td><td>Only serious errors or worse</td></tr>
<tr><td><code>warn</code></td><td>Only warnings and errors are logged</td></tr>
</table>
</td></tr>
<tr><td><code>flush_time</code></td><td>How long to wait before flushing the request log files for each virtual server.

Value type: <code>UInt</code>

Default: <code>5</code></td></tr>
<tr><td><code>log_file</code></td><td>The file to log event messages to.

Value type: <code>String</code>

Default: <code>%zeushome%/zxtm/log/errors</code></td></tr>
<tr><td><code>rate</code></td><td>The maximum number of connection errors logged per second when connection error reporting is enabled.

Value type: <code>UInt</code>

Default: <code>50</code></td></tr>
<tr><td><code>reopen</code></td><td>How long to wait before re-opening request log files, this ensures that log files will be recreated in the case of log rotation.

Value type: <code>UInt</code>

Default: <code>30</code></td></tr>
<tr><td><code>time</code></td><td>The minimum time between log messages for log intensive features such as SLM.

Value type: <code>UInt</code>

Default: <code>60</code></td></tr>
<tr><td colspan="2">Properties for the <code>periodic_log</code> section:</td></tr>
<tr><td><code>enabled</code></td><td>Enable periodic logging</td></tr>
</table>

	Value type: `Boolean` Default: `true`
`interval`	Time interval in seconds for periodic logging Value type: `UInt` Default: `600`
`max_archive_set_size`	Maximum size (in MBytes) for the archive periodic logs. When combined size of the archives exceeds this value, the oldest archives will be deleted. Set to 0 to disable archive size limit Value type: `UInt` Default: `50`
`max_log_set_size`	Maximum size (in MBytes) for the current set of periodic logs. If this size is exceeded, the current set will be archived. Set to zero to disable archiving based on current set size. Value type: `UInt` Default: `25`
`max_num_archives`	Maximum number of archived log sets to keep. When the number of archived periodic log sets exceeds this, the oldest archives will be deleted. Value type: `UInt` Default: `14`
`run_count`	Number of periodic logs which should be archived together as a run. Value type: `UInt` Default: `144`
Properties for the `recent_connections` section:	
`max_per_process`	How many recently closed connections each traffic manager process should save. These saved connections will be shown alongside currently active connections when viewing the Connections page. You should set this value to `0` in a benchmarking or performance-critical environment. Value type: `UInt`

	Default: `<none>`
`retain_time`	The amount of time for which snapshots will be retained on the Connections page. Value type: `UInt` Default: `60`
`snapshot_size`	The maximum number of connections each traffic manager process should show when viewing a snapshot on the Connections page. This value includes both currently active connections and saved connections. If set to `0` all active and saved connection will be displayed on the Connections page. Value type: `UInt` Default: `500`
Properties for the `rest_api` section:	
`auth_timeout`	The length of time after a successful request that the authentication of a given username and password will be cached for an IP address. A setting of 0 disables the cache forcing every REST request to be authenticated which will adversely affect performance. Value type: `UInt` Default: `120`
`enabled`	Whether or not the REST service is enabled. Value type: `Boolean` Default: `false`
`http_max_header_length`	The maximum allowed length in bytes of a HTTP request's headers. Value type: `UInt` Default: `4096`
`proxy_map`	A set of symlinks that the REST API maps to actual directories. Used to add mirored resources so proxies work correctly. primary key: `absolute_path (String)` — The real path to create a symlinked resource to.

<table>
<tr><td></td><td>sub keys:<table><tr><td><code>symlink_path (String)</code></td><td>The path to the symlinked resource. Intermediate resources will be created. All new resources will be hidden.</td></tr></table></td></tr>
<tr><td><code>replicate_absolute</code></td><td>Configuration changes will be replicated across the cluster after this period of time, regardless of whether additional API requests are being made.

Value type: <code>UInt</code>

Default: <code>20</code></td></tr>
<tr><td><code>replicate_lull</code></td><td>Configuration changes made via the REST API will be propagated across the cluster when no further API requests have been made for this period of time.

Value type: <code>UInt</code>

Default: <code>5</code></td></tr>
<tr><td><code>replicate_timeout</code></td><td>The period of time after which configuration replication across the cluster will be cancelled if it has not completed.

Value type: <code>UInt</code>

Default: <code>10</code></td></tr>
<tr><td colspan="2">Properties for the <code>security</code> section:</td></tr>
<tr><td><code>login_banner</code></td><td>Banner text displayed on the Admin Server login page and before logging in to appliance SSH servers.

Value type: <code>FreeformString</code>

Default: <code><none></code></td></tr>
<tr><td><code>login_banner_accept</code></td><td>Whether or not users must explicitly agree to the displayed <code>login_banner</code> text before logging in to the Admin Server.

Value type: <code>Boolean</code>

Default: <code>false</code></td></tr>
<tr><td><code>login_delay</code></td><td>The number of seconds before another login attempt can be made after a failed attempt.

Value type: <code>UInt</code></td></tr>
</table>

	Default: <none>
max_login_attempts	The number of sequential failed login attempts that will cause a user account to be suspended. Setting this to 0 disables this feature. To apply this to users who have never successfully logged in, track_unknown_users must also be enabled. Value type: UInt Default: <none>
max_login_external	Whether or not usernames blocked due to the max_login_attempts limit should also be blocked from authentication against external services (such as LDAP and RADIUS). Value type: Boolean Default: false
max_login_suspension_time	The number of minutes to suspend users who have exceeded the max_login_attempts limit. Value type: UInt Default: 15
password_allow_consecutive_chars	Whether or not to allow the same character to appear consecutively in passwords. Value type: Boolean Default: true
password_changes_per_day	The maximum number of times a password can be changed in a 24-hour period. Set to 0 to disable this restriction. Value type: UInt Default: <none>
password_min_alpha_chars	Minimum number of alphabetic characters a password must contain. Set to 0 to disable this restriction. Value type: UInt Default: <none>
password_min_length	Minimum number of characters a password must contain. Set to 0 to

	disable this restriction. Value type: `UInt` Default: `<none>`
`password_min_numeric_chars`	Minimum number of numeric characters a password must contain. Set to `0` to disable this restriction. Value type: `UInt` Default: `<none>`
`password_min_special_chars`	Minimum number of special (non-alphanumeric) characters a password must contain. Set to `0` to disable this restriction. Value type: `UInt` Default: `<none>`
`password_min_uppercase_chars`	Minimum number of uppercase characters a password must contain. Set to `0` to disable this restriction. Value type: `UInt` Default: `<none>`
`password_reuse_after`	The number of times a password must have been changed before it can be reused. Set to `0` to disable this restriction. Value type: `UInt` Default: `<none>`
`post_login_banner`	Banner text to be displayed on the appliance console after login. Value type: `String` Default: `<none>`
`track_unknown_users`	Whether to remember past login attempts from usernames that are not known to exist (should be set to false for an Admin Server accessible from the public Internet). This does not affect the audit log. Value type: `Boolean` Default: `false`

ui_page_banner	Banner text to be displayed on all Admin Server pages. Value type: String Default: <none>
Properties for the session section:	
asp_cache_size	The maximum number of entries in the ASP session cache. This is used for storing session mappings for ASP session persistence. Approximately 100 bytes will be pre-allocated per entry. Value type: UInt Default: 2048
ip_cache_size	The maximum number of entries in the IP session cache. This is used to provide session persistence based on the source IP address. Approximately 100 bytes will be pre-allocated per entry. Value type: UInt Default: 2048
j2ee_cache_size	The maximum number of entries in the J2EE session cache. This is used for storing session mappings for J2EE session persistence. Approximately 100 bytes will be pre-allocated per entry. Value type: UInt Default: 2048
ssl_cache_size	The maximum number of entries in the SSL session persistence cache. This is used to provide session persistence based on the SSL session ID. Approximately 200 bytes will be pre-allocated per entry. Value type: UInt Default: 2048
universal_cache_size	The maximum number of entries in the global universal session cache. This is used for storing session mappings for universal session persistence. Approximately 100 bytes will be pre-allocated per entry. Value type: UInt Default: 2048
Properties for the snmp section:	

user_counters	The number of user defined SNMP counters. Value type: UInt Default: 10
Properties for the soap section:	
idle_minutes	The number of minutes that the SOAP server should remain idle before exiting. The SOAP server has a short startup delay the first time a SOAP request is made, subsequent SOAP requests don't have this delay. Value type: UInt Default: 10
Properties for the ssl section:	
cache_expiry	How long the SSL session IDs for SSL decryption should be stored for. Value type: UInt Default: 1800
cache_size	How many entries the SSL session ID cache should hold. This cache is used to cache SSL sessions to help speed up SSL handshakes when performing SSL decryption. Each entry will allocate approximately 1.5kB of metadata. Value type: UInt Default: 6151
crl_mem_size	How much shared memory to allocate for loading Certificate Revocation Lists. This should be at least 3 times the total size of all CRLs on disk. This is specified as either a percentage of system RAM, 1% for example, or an absolute size such as 10MB. Value type: String Default: 5MB
insert_extra_fragment	Whether or not SSL3 and TLS1 use one-byte fragments as a BEAST countermeasure. Value type: Boolean

	Default: `false`
`max_handshake_message_size`	The maximum size (in bytes) of SSL handshake messages that SSL connections will accept. To accept any size of handshake message the key should be set to the value `0`. Value type: `UInt` Default: `10240`
`ocsp_cache_size`	The maximum number of cached client certificate OCSP results stored. This cache is used to speed up OCSP checks against client certificates by caching results. Approximately 1040 bytes are pre-allocated per entry. Value type: `UInt` Default: `2048`
`ocsp_stapling_default_refresh_interval`	How long to wait before refreshing requests on behalf of the store of certificate status responses used by OCSP stapling, if we don't have an up-to-date OCSP response. Value type: `UInt` Default: `60`
`ocsp_stapling_mem_size`	How much shared memory to allocate for the store of certificate status responses for OCSP stapling. This should be at least 2kB times the number of certificates configured to use OCSP stapling. This is specified as either a percentage of system RAM, `1%` for example, or an absolute size such as `10MB`. Value type: `String` Default: `1MB`
`ocsp_stapling_minimum_refresh_interval`	The minimum number of seconds to wait between OCSP requests for the same certificate. Value type: `UInt` Default: `10`
`ocsp_stapling_prefetch`	The number of seconds before an OCSP response is stale to make a new OCSP request. Value type: `UInt`

<table>
<tr><td></td><td>Default: 30</td></tr>
<tr><td>prevent_timing_side_channels</td><td>Take performance degrading steps to prevent exposing timing side-channels with SSL3 and TLS.

Value type: Boolean

Default: false</td></tr>
<tr><td>ssl3_allow_rehandshake</td><td>Whether or not SSL3/TLS re-handshakes should be supported. Enabling support for re-handshakes can expose services to Man-in-the-Middle attacks. It is recommended that only "safe" handshakes be permitted, or none at all.

Value type: Enum(String)

Default: safe

Permitted values:<table><tr><td>always</td><td>Always allow</td></tr><tr><td>never</td><td>Never allow</td></tr><tr><td>rfc5746</td><td>Only if client uses RFC 5746 (Secure Renegotiation Extension)</td></tr><tr><td>safe</td><td>Allow safe re-handshakes</td></tr></table></td></tr>
<tr><td>ssl3_ciphers</td><td>The SSL ciphers to use. For information on supported ciphers see the online help.

Value type: String

Default: <none></td></tr>
<tr><td>ssl3_diffie_hellman_key_length</td><td>The length in bits of the Diffie-Hellman key for ciphers that use Diffie-Hellman key agreement.

Value type: Enum(UInt)

Default: dh_1024

Permitted values:<table><tr><td>dh_1024</td><td>1024</td></tr><tr><td>dh_2048</td><td>2048</td></tr><tr><td>dh_3072</td><td>3072</td></tr><tr><td>dh_4096</td><td>4096</td></tr></table></td></tr>
<tr><td>ssl3_min_rehandshake</td><td>If SSL3/TLS re-handshakes are supported, this defines the minimum time interval (in milliseconds) between handshakes on a single</td></tr>
</table>

_interval	SSL3/TLS connection that is permitted. To disable the minimum interval for handshakes the key should be set to the value 0. Value type: UInt Default: 1000
support_ssl2	Whether or not SSL2 support is enabled. Value type: Boolean Default: false
support_ssl3	Whether or not SSL3 support is enabled. Value type: Boolean Default: true
support_tls1	Whether or not TLS1.0 support is enabled. Value type: Boolean Default: true
support_tls1_1	Whether or not TLS1.1 support is enabled. Value type: Boolean Default: true
Properties for the ssl_hardware section:	
accel	Whether or not the SSL hardware is an "accelerator" (faster than software). By default the traffic manager will only use the SSL hardware if a key requires it (i.e. the key is stored on secure hardware and the traffic manager only has a placeholder/identifier key). With this option enabled, your traffic manager will instead try to use hardware for all SSL decrypts. Value type: Boolean Default: false
driver_pkcs11_debug	Print verbose information about the PKCS11 hardware security module to the event log. Value type: Boolean

<table>
<tr><td></td><td>Default: <code>false</code></td></tr>
<tr><td><code>driver_pkcs11_lib</code></td><td>The location of the PKCS#11 library for your SSL hardware if it is not in a standard location. The traffic manager will search the standard locations by default.
Value type: <code>String</code>
Default: <code><none></code></td></tr>
<tr><td><code>driver_pkcs11_slot_desc</code></td><td>The label of the SSL Hardware slot to use. Only required if you have multiple HW accelerator slots.
Value type: <code>String</code>
Default: <code><none></code></td></tr>
<tr><td><code>driver_pkcs11_slot_type</code></td><td>The type of SSL hardware slot to use.
Value type: <code>Enum(String)</code>
Default: <code>operator</code>
Permitted values:
<table>
<tr><td><code>module</code></td><td>Local Module</td></tr>
<tr><td><code>operator</code></td><td>Operator Card Set</td></tr>
<tr><td><code>softcard</code></td><td>Soft Card</td></tr>
</table></td></tr>
<tr><td><code>driver_pkcs11_user_pin</code></td><td>The User PIN for the PKCS token (PKCS#11 devices only).
Value type: <code>Password</code>
Default: <code><none></code></td></tr>
<tr><td><code>failure_count</code></td><td>The number of consecutive failures from the SSL hardware that will be tolerated before the traffic manager assumes its session with the device is invalid and tries to log in again. This is necessary when the device reboots following a power failure.
Value type: <code>UInt</code>
Default: <code>5</code></td></tr>
<tr><td><code>library</code></td><td>The type of SSL hardware to use. The drivers for the SSL hardware should be installed and accessible to the traffic manager software.
Value type: <code>Enum(String)</code></td></tr>
</table>

<table>
<tr><td></td><td>Default: <code>none</code>

Permitted values:
<table>
<tr><td><code>cn1000</code></td><td>Cavium Networks CN1000</td></tr>
<tr><td><code>cn2000</code></td><td>Cavium Networks CN2000</td></tr>
<tr><td><code>none</code></td><td>None</td></tr>
<tr><td><code>pkcs11</code></td><td>PKCS#11 (e.g. nCipher NetHSM, Sun SCA 6000)</td></tr>
</table>
</td></tr>
<tr><td colspan="2">Properties for the <code>trafficscript</code> section:</td></tr>
<tr><td><code>array_elements</code></td><td>The amount of storage that will be allocated to array elements in TrafficScript. If more elements are required then the necessary memory will be allocated during the execution of the rule.

Value type: <code>UInt</code>

Default: <code>100000</code></td></tr>
<tr><td><code>data_local_size</code></td><td>The maximum amount of memory available to store TrafficScript <code>data.local.set()</code> information. This can be specified as a percentage of system RAM, <code>5%</code> for example; or an absolute size such as <code>200MB</code>.

Value type: <code>String</code>

Default: <code>5%</code></td></tr>
<tr><td><code>data_size</code></td><td>The maximum amount of memory available to store TrafficScript <code>data.set()</code> information. This can be specified as a percentage of system RAM, <code>5%</code> for example; or an absolute size such as <code>200MB</code>.

Value type: <code>String</code>

Default: <code>5%</code></td></tr>
<tr><td><code>max_instr</code></td><td>The maximum number of instructions a TrafficScript rule will run. A rule will be aborted if it runs more than this number of instructions without yielding, preventing infinite loops.

Value type: <code>UInt</code>

Default: <code>100000</code></td></tr>
<tr><td><code>memory_warning</code></td><td>Raise an event if a TrafficScript rule requires more than this amount of buffered network data. If you get such events repeatedly, you may want to consider re-working some of your TrafficScript rules to use less memory or to stream the data that they process rather than</td></tr>
</table>

	storing it all in memory. This setting also limits the amount of data that can be returned by `request.GetLine()`. Value type: `UInt` Default: `1048576`
`regex_cache_size`	The maximum number of regular expressions to cache in TrafficScript. Regular expressions will be compiled in order to speed up their use in the future. Value type: `UInt` Default: `57`
`regex_match_limit`	The maximum number of ways TrafficScript will attempt to match a regular expression at each position in the subject string, before it aborts the rule and reports a TrafficScript error. Value type: `UInt` Default: `10000000`
`regex_match_warn_percentage`	The percentage of `regex_match_limit` at which TrafficScript reports a performance warning. Value type: `UInt` Default: `5`
`variable_pool_use`	Allow the `pool.use` and `pool.select` TrafficScript functions to accept variables instead of requiring literal strings. Enabling this feature has the following effects 1. Your traffic manager may no longer be able to know whether a pool is in use. 2. Errors for pools that aren't in use will not be hidden. 3. Some settings displayed for a Pool may not be appropriate for the type of traffic being managed. 4. Pool usage information on the pool edit pages and config summary may not be accurate. 5. Monitors will run for all pools (with this option disabled monitors will only run for Pools that are used). Value type: `Boolean` Default: `false`
Properties for the `web_cache` section:	

avg_path_length	The estimated average length of the path (including query string) for resources being cached. An amount of memory equal to this figure multiplied by max_file_num will be allocated for storing the paths for cache entries. This setting can be increased if your web site makes extensive use of long URLs. Value type: UInt Default: 512
disk	Whether or not to use a disk-backed (typically SSD) cache. If set to Yes cached web pages will be stored in a file on disk. This enables the traffic manager to use a cache that is larger than available RAM. The size setting should also be adjusted to select a suitable maximum size based on your disk space. Note that the disk caching is optimized for use with SSD storage. Value type: Boolean Default: false
disk_dir	If disk caching is enabled, this sets the directory where the disk cache file will be stored. The traffic manager will create a file called webcache.data in this location. Note that the disk caching is optimized for use with SSD storage. Value type: String Default: %zeushome%/zxtm/internal
max_file_num	Maximum number of entries in the cache. Approximately 0.9 KB will be pre-allocated per entry for metadata, this is in addition to the memory reserved for the content cache and for storing the paths of the cached resources. Value type: UInt Default: 10000
max_file_size	Largest size of a cacheable object in the cache. This is specified as either a percentage of the total cache size, 2% for example, or an absolute size such as 20MB. Value type: String Default: 2%
max_path_length	The maximum length of the path (including query string) for the resource being cached. If the path exceeds this length then it will not

	be added to the cache. Value type: `UInt` Default: `2048`
`normalize_query`	Enable normalization (lexical ordering of the parameter-assignments) of the query string. Value type: `Boolean` Default: `true`
`size`	The maximum size of the HTTP web page cache. This is specified as either a percentage of system RAM, `20%` for example, or an absolute size such as `200MB`. Value type: `String` Default: `20%`
`verbose`	Add an X-Cache-Info header to every HTTP response, showing whether the request and/or the response was cacheable. Value type: `Boolean` Default: `false`

License

URI Path: `license_keys`

A license key is a encoded text file that controls what functionality is available from each traffic manager in the cluster. Every production traffic manager must have a valid licence key in order to function; a traffic manager without a license will operate in developer mode, allowing developers to trial a wide range of functionality, but placing restrictions on bandwidth.

Property	Description
There are no properties to display for this resource.	

Location

URI Path: `locations`

These are geographic locations as used by **Global Load Balancing** services. Such a location may not necessarily contain a traffic manager; instead it could refer to the location of a remote datacenter.

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>id</td><td>The identifier of this location.
Value type: UInt
Default: <none></td></tr>
<tr><td>latitude</td><td>The latitude of this location.
Value type: Float
Default: 0.0</td></tr>
<tr><td>longitude</td><td>The longitude of this location.
Value type: Float
Default: 0.0</td></tr>
<tr><td>note</td><td>A note, used to describe this location.
Value type: FreeformString
Default: <none></td></tr>
<tr><td>type</td><td>Does this location contain traffic managers and configuration or is it a recipient of GLB requests?
Value type: Enum(String)
Default: config
Permitted values:<table><tr><td>config</td><td>Configuration</td></tr><tr><td>glb</td><td>GLB</td></tr></table></td></tr>
</table>

Monitor

URI Path: `monitors`

Monitors check important remote services are running, by periodically sending them traffic and checking the response is correct. They are used by virtual servers to detect the failure of backend nodes.

Property	Description
`back_off`	Should the monitor slowly increase the delay after it has failed? Value type: `Boolean` Default: `true`
`delay`	The minimum time between calls to a monitor. Value type: `UInt` Default: `3`
`failures`	The number of times in a row that a node must fail execution of the monitor before it is classed as unavailable. Value type: `UInt` Default: `3`
`machine`	The machine to monitor, where relevant this should be in the form `<hostname>:<port>`, for "ping" monitors the `:<port>` part must not be specified. Value type: `String` Default: `<none>`
`note`	A description of the montitor. Value type: `FreeformString` Default: `<none>`
`scope`	A monitor can either monitor each node in the pool separately and disable an individual node if it fails, or it can monitor a specific machine and disable the entire pool if that machine fails. GLB location monitors must monitor a specific machine. Value type: `Enum(String)`

<table>
<tr><td></td><td>Default: <code>pernode</code>

Permitted values:
<table>
<tr><td><code>pernode</code></td><td>Node: Monitor each node in the pool separately</td></tr>
<tr><td><code>poolwide</code></td><td>Pool/GLB: Monitor a specified machine</td></tr>
</table>
</td></tr>
<tr><td><code>timeout</code></td><td>The maximum runtime for an individual instance of the monitor.

Value type: <code>UInt</code>

Default: <code>3</code></td></tr>
<tr><td><code>type</code></td><td>The internal monitor implementation of this monitor.

Value type: <code>Enum(String)</code>

Default: <code>ping</code>

Permitted values:
<table>
<tr><td><code>connect</code></td><td>TCP Connect monitor</td></tr>
<tr><td><code>http</code></td><td>HTTP monitor</td></tr>
<tr><td><code>ping</code></td><td>Ping monitor</td></tr>
<tr><td><code>program</code></td><td>External program monitor</td></tr>
<tr><td><code>rtsp</code></td><td>RTSP monitor</td></tr>
<tr><td><code>sip</code></td><td>SIP monitor</td></tr>
<tr><td><code>tcp_transaction</code></td><td>TCP transaction monitor</td></tr>
</table>
</td></tr>
<tr><td><code>use_ssl</code></td><td>Whether or not the monitor should connect using SSL.

Value type: <code>Boolean</code>

Default: <code>false</code></td></tr>
<tr><td><code>verbose</code></td><td>Whether or not the monitor should emit verbose logging. This is useful for diagnosing problems.

Value type: <code>Boolean</code>

Default: <code>false</code></td></tr>
<tr><td colspan="2">Properties for the <code>http</code> section:</td></tr>
<tr><td><code>authentication</code></td><td>The HTTP basic-auth <code><user>:<password></code> to use for the test</td></tr>
</table>

	HTTP request. Value type: `String` Default: `<none>`
`body_regex`	A regular expression that the HTTP response body must match. If the response body content doesn't matter then set this to `.*` (match anything). Value type: `String` Default: `<none>`
`host_header`	The host header to use in the test HTTP request. Value type: `String` Default: `<none>`
`path`	The path to use in the test HTTP request. This must be a string beginning with a / (forward slash). Value type: `String` Default: `/`
`status_regex`	A regular expression that the HTTP status code must match. If the status code doesn't matter then set this to `.*` (match anything). Value type: `String` Default: `^[234][0-9][0-9]$`
Properties for the `rtsp` section:	
`body_regex`	The regular expression that the RTSP response body must match. Value type: `String` Default: `<none>`
`path`	The path to use in the RTSP request (some servers will return 500 Internal Server Error unless this is a valid media file). Value type: `String` Default: `/`

<table>
<tr><td>status_regex</td><td>The regular expression that the RTSP response status code must match.

Value type: String

Default: ^[234][0-9][0-9]$</td></tr>
<tr><td colspan="2">Properties for the script section:</td></tr>
<tr><td>arguments</td><td>A table containing arguments and argument values to be passed to the monitor program.

primary key:<table><tr><td>name (String)</td><td>The name of the argument to be passed to the monitor program.</td></tr></table>
sub keys:<table><tr><td>value (String)</td><td>The value of the argument to be passed to the monitor program.</td></tr><tr><td>description (String)</td><td>A description for the argument provided to the program.</td></tr></table></td></tr>
<tr><td>program</td><td>The program to run. This must be an executable file, either within the monitor scripts directory or specified as an absolute path to some other location on the filesystem.

Value type: String

Default: <none></td></tr>
<tr><td colspan="2">Properties for the sip section:</td></tr>
<tr><td>body_regex</td><td>The regular expression that the SIP response body must match.

Value type: String

Default: <none></td></tr>
<tr><td>status_regex</td><td>The regular expression that the SIP response status code must match.

Value type: String

Default: ^[234][0-9][0-9]$</td></tr>
<tr><td>transport</td><td>Which transport protocol the SIP monitor will use to query the server.

Value type: Enum(String)</td></tr>
</table>

<table>
<tr><td></td><td>Default: udp

Permitted values:
<table>
<tr><td>tcp</td><td>TCP</td></tr>
<tr><td>udp</td><td>UDP</td></tr>
</table>
</td></tr>
<tr><td colspan="2">Properties for the tcp section:</td></tr>
<tr><td>close_string</td><td>An optional string to write to the server before closing the connection.

Value type: String

Default: <none></td></tr>
<tr><td>max_response_len</td><td>The maximum amount of data to read back from a server, use 0 for unlimited. Applies to TCP and HTTP monitors.

Value type: UInt

Default: 2048</td></tr>
<tr><td>response_regex</td><td>A regular expression to match against the response from the server. Applies to TCP monitors only.

Value type: String

Default: .+</td></tr>
<tr><td>write_string</td><td>The string to write down the TCP connection.

Value type: String

Default: <none></td></tr>
<tr><td colspan="2">Properties for the udp section:</td></tr>
<tr><td>accept_all</td><td>If this monitor uses UDP, should it accept responses from any IP and port?

Value type: Boolean

Default: false</td></tr>
</table>

Monitor Program

URI Path: `monitor_scripts`

An executable program that can be used to by external program monitors to report the health of backend services.

Property	Description
There are no properties to display for this resource.	

Pool

URI Path: `pools`

A pool manages a group of backend nodes. It routes traffic to the most appropriate node, based on load balancing and session persistence criteria.

Property	Description
`bandwidth_class`	The Bandwidth Management Class this pool uses, if any. Value type: `Reference(config-bandwidth)` Default: `<none>`
`disabled`	A list of nodes in the pool that are in the 'disabled' state. Value type: `Set(String)` Default: `<none>`
`draining`	A list of nodes in the pool that are in the 'draining' state. Value type: `Set(String)` Default: `<none>`
`failure_pool`	If all of the nodes in this pool have failed, then requests can be diverted to another pool. Value type: `Reference(config-pool)`

	Default: `<none>`
`max_connection_attempts`	The maximum number of nodes to which the traffic manager will attempt to send a request before returning an error to the client. Requests that are non-retryable will be attempted against only one node. Zero signifies no limit. Value type: `UInt` Default: `<none>`
`max_idle_connections_pernode`	The maximum number of unused HTTP keepalive connections that should be maintained to an individual node. Zero signifies no limit. Value type: `UInt` Default: `50`
`max_timed_out_connection_attempts`	The maximum number of connection attempts the traffic manager will make where the server fails to respond within the time limit defined by the `max_reply_time` setting. Zero signifies no limit. Value type: `UInt` Default: `2`
`monitors`	The monitors assigned to this pool, used to detect failures in the back end nodes. Value type: `Set(Reference(config-monitor))` Default: `<none>`
`node_connection_attempts`	The number of times the software will attempt to connect to the same back-end node before marking it as failed. This is only used when `passive_monitoring` is enabled. Value type: `UInt` Default: `3`

nodes	A list of all active and draining nodes in this pool. A node should be specified as a <ip>:<port> pair. Value type: Set(String) Default: <none>
note	A description of the pool. Value type: String Default: <none>
passive_monitoring	Whether or not the software should check that 'real' requests (i.e. not those from monitors) to this pool appear to be working. This should normally be enabled, so that when a node is refusing connections, responding too slowly, or sending back invalid data, it can mark that node as failed, and stop sending requests to it. If this is disabled, you should ensure that suitable health monitors are configured to check your servers instead, otherwise failed requests will not be detected and subsequently retried. Value type: Boolean Default: true
persistence_class	The default Session Persistence class this pool uses, if any. Value type: Reference(config-persistence) Default: <none>
transparent	Whether or not connections to the back-ends appear to originate from the source client IP address. Value type: Boolean Default: false
Properties for the auto_scaling section:	

cloud_credentials	The Cloud Credentials object containing authentication credentials to use in cloud API calls. Value type: Reference(cloud-api) Default: <none>
cluster	The ESX host or ESX cluster name to put the new virtual machine instances on. Value type: String Default: <none>
data_center	The name of the logical datacenter on the vCenter server. Virtual machines will be scaled up and down under the datacenter root folder. Value type: String Default: <none>
data_store	The name of the datastore to be used by the newly created virtual machine. Value type: String Default: <none>
enabled	Are the nodes of this pool subject to auto-scaling? If yes, nodes will be automatically added and removed from the pool by the chosen auto-scaling mechanism. Value type: Boolean Default: false
external	Whether or not auto-scaling is being handled by an external system. Set this value to true if all aspects of auto-scaling are handled by an external system, such as RightScale. If set to false, the traffic manager will determine when to scale the pool and will communicate with the cloud provider to create and destroy nodes as necessary.

<table>
<tr><td></td><td>Value type: <code>Boolean</code>

Default: <code>true</code></td></tr>
<tr><td><code>hysteresis</code></td><td>The time period in seconds for which a change condition must persist before the change is actually instigated.

Value type: <code>UInt</code>

Default: <code>20</code></td></tr>
<tr><td><code>imageid</code></td><td>The identifier for the image of the instances to create.

Value type: <code>String</code>

Default: <code><none></code></td></tr>
<tr><td><code>ips_to_use</code></td><td>Which type of IP addresses on the node to use. Choose private IPs if the traffic manager is in the same cloud as the nodes, otherwise choose public IPs.

Value type: <code>Enum(String)</code>

Default: <code>publicips</code>

Permitted values:<table><tr><td><code>private_ips</code></td><td>Private IP addresses</td></tr><tr><td><code>publicips</code></td><td>Public IP addresses</td></tr></table></td></tr>
<tr><td><code>last_node_idle_time</code></td><td>The time in seconds for which the last node in an auto-scaled pool must have been idle before it is destroyed. This is only relevant if min_nodes is 0.

Value type: <code>UInt</code>

Default: <code>3600</code></td></tr>
<tr><td><code>max_nodes</code></td><td>The maximum number of nodes in this auto-scaled pool.

Value type: <code>UInt</code>

Default: <code>4</code></td></tr>
</table>

min_nodes	The minimum number of nodes in this auto-scaled pool. Value type: UInt Default: 1
name	The beginning of the name of nodes in the cloud that are part of this auto-scaled pool. Value type: String Default: <none>
port	The port number to use for each node in this auto-scaled pool. Value type: UInt Default: 80
refractory	The time period in seconds after the instigation of a re-size during which no further changes will be made to the pool size. Value type: UInt Default: 180
response_time	The expected response time of the nodes in ms. This time is used as a reference when deciding whether a node's response time is conforming. All responses from all the nodes will be compared to this reference and the percentage of conforming responses is the base for decisions about scaling the pool up or down. Value type: UInt Default: 1000
scale_down_level	The fraction, in percent, of conforming requests above which the pool size is decreased. If the percentage of conforming requests exceeds this value, the pool is scaled down.

	Value type: `UInt` Default: `95`
`scale_up_level`	The fraction, in percent, of conforming requests below which the pool size is increased. If the percentage of conforming requests drops below this value, the pool is scaled up. Value type: `UInt` Default: `40`
`size_id`	The identifier for the size of the instances to create. Value type: `String` Default: `<none>`
Properties for the `connection` section:	
`max_connect_time`	How long the pool should wait for a connection to a node to be established before giving up and trying another node. Value type: `UInt` Default: `4`
`max_connections_per_node`	The maximum number of concurrent connections allowed to each back-end node in this pool per machine. A value of 0 means unlimited connections. Value type: `UInt` Default: `<none>`
`max_queue_size`	The maximum number of connections that can be queued due to connections limits. A value of 0 means unlimited queue size. Value type: `UInt`

	Default: `<none>`
`max_reply_time`	How long the pool should wait for a response from the node before either discarding the request or trying another node (retryable requests only). Value type: `UInt` Default: `30`
`queue_timeout`	The maximum time to keep a connection queued in seconds. Value type: `UInt` Default: `10`
Properties for the `ftp` section:	
`support_rfc_2428`	Whether or not the backend IPv4 nodes understand the EPRT and EPSV command from RFC 2428. It is always assumed that IPv6 nodes support these commands. Value type: `Boolean` Default: `false`
Properties for the `http` section:	
`keepalive`	Whether or not the pool should maintain HTTP keepalive connections to the nodes. Value type: `Boolean` Default: `true`
`keepalive_non_idempotent`	Whether or not the pool should maintain HTTP keepalive connections to the nodes for non-idempotent requests. Value type: `Boolean` Default: `false`

<table>
<tr><th colspan="3">Properties for the <code>load_balancing</code> section:</th></tr>
<tr><td><code>algorithm</code></td><td colspan="2">The load balancing algorithm that this pool uses to distribute load across its nodes.

Value type: <code>Enum(String)</code>

Default: <code>round_robin</code>

Permitted values:</td></tr>
<tr><td></td><td><code>fastest_response_time</code></td><td>The Response Time algorithm monitors the response times for recent requests to each node. It sends each new request to the node that has recently been responding the most quickly.</td></tr>
<tr><td></td><td><code>least_connections</code></td><td>This algorithm sends each new request to the node with the fewest currently active connections.</td></tr>
<tr><td></td><td><code>perceptive</code></td><td>The Perceptive algorithm uses a combination of</td></tr>
</table>

			response time data and connection counts to predict which node is likely to have the fastest response time for each request.
		`random`	This algorithm chooses a random node for each request.
		`round_robin`	This algorithm distributes traffic by assigning each request to a new node in turn.
		`weighted_least_connections`	This algorithm works in a similar way to the Least Connections algorithm, but assigns more requests to nodes with a greater 'weight'.

<table>
<tr><td></td><td><table><tr><td>weighted_round_robin</td><td>Weighted Round Robin works in a similar way to Round Robin, but assigns more requests to nodes with a greater 'weight'.</td></tr></table></td></tr>
<tr><td>node_weighting</td><td>A table containing per-node weighting for use in some load balancing algorithms (weighted least connections and weighted round robin).
primary key:<table><tr><td>node (String)</td><td>Node to which the weighting should be applied.</td></tr></table>sub keys:<table><tr><td>weight (Int)</td><td>Weight for the node. The actual value in isolation does not matter: As long as it is a valid integer 1-100, the per-node weightings are calculated on the relative values between the nodes.</td></tr></table></td></tr>
<tr><td>priority_enabled</td><td>Enable priority lists.
Value type: Boolean
Default: false</td></tr>
<tr><td>priority_nodes</td><td>Minimum number of highest-priority active nodes.
Value type: UInt
Default: 1</td></tr>
<tr><td>priority_values</td><td>A list of node priorities, higher values signify higher priority. Priorities are specified using the format <ip>:<port>:<priority>, if a priority is not specified for a node it is assumed to be 1.</td></tr>
</table>

	Value type: `Set(String)` Default: `<none>`
Properties for the `node` section:	
`close_on_death`	Close all connections to a node once we detect that it has failed. Value type: `Boolean` Default: `false`
`retry_fail_time`	The amount of time, in seconds, that a traffic manager will wait before re-trying a node that has been marked as failed by passive monitoring. Value type: `UInt` Default: `60`
Properties for the `smtp` section:	
`send_starttls`	If we are encrypting traffic for an SMTP connection, should we upgrade to SSL using STARTTLS. Value type: `Boolean` Default: `true`
Properties for the `ssl` section:	
`client_auth`	Whether or not a suitable certificate and private key from the SSL Client Certificates catalog be used if the back-end server requests client authentication. Value type: `Boolean` Default: `false`
`enable`	Whether or not the pool should encrypt data before sending it to a back-end node. Value `Boolean`

	type: Default: `false`
`enhance`	SSL protocol enhancements allow your traffic manager to prefix each new SSL connection with information about the client. This enables Riverbed Web Servers to run multiple SSL sites, and to discover the client's IP address. Only enable this if you are using nodes for this pool which are Riverbed Web Servers or Stingray Traffic Managers, whose virtual servers have the `ssl_trust_magic` setting enabled. Value type: `Boolean` Default: `false`
`send_close_alerts`	Whether or not to send an SSL/TLS "close alert" when initiating a socket disconnection. Value type: `Boolean` Default: `false`
`server_name`	Whether or not the software should use the TLS 1.0 server_name extension, which may help the back-end node provide the correct certificate. Enabling this setting will force the use of at least TLS 1.0. Value type: `Boolean` Default: `false`
`strict_verify`	Whether or not strict certificate verification should be performed. This will turn on checks to disallow server certificates that don't match the server name, are self-signed, expired, revoked, or have an unknown CA. Value type: `Boolean` Default: `false`
Properties for the `tcp` section:	

<table>
<tr><td>nagle</td><td>Whether or not Nagle's algorithm should be used for TCP connections to the back-end nodes.

Value type: Boolean

Default: true</td></tr>
<tr><td colspan="2">Properties for the udp section:</td></tr>
<tr><td>accept_from</td><td>The IP addresses and ports from which responses to UDP requests should be accepted.
If set to accept responses from a specific set of IP addresses, you will need to enter a CIDR Mask (such as 10.100.0.0/16).

Value type: Enum(String)

Default: dest_only

Permitted values:
<table>
<tr><td>all</td><td>Any IP address and any port.</td></tr>
<tr><td>dest_ip_only</td><td>Only the IP address to which the request was sent, but from any port.</td></tr>
<tr><td>dest_only</td><td>Only the IP address and port to which the request was sent.</td></tr>
<tr><td>ip_mask</td><td>Only a specific set of IP addresses, but from any port.</td></tr>
</table>
</td></tr>
<tr><td>accept_from_mask</td><td>The CIDR mask that matches IPs we want to receive responses from.

Value type: String

Default: <none></td></tr>
</table>

Protection Class

URI Path: `protection`

A protection class specifies the level of protection against network attacks for a virtual server.

Property	Description
`debug`	Whether or not to output verbose logging. Value type: `Boolean` Default: `false`
`enabled`	Enable or disable this service protection class. Value type: `Boolean` Default: `true`
`log_time`	Log service protection messages at these intervals. If set to `0` no messages will be logged and no alerts will be sent. Value type: `UInt` Default: `60`
`note`	A description of the service protection class. Value type: `String` Default: `<none>`
`rule`	A TrafficScript rule that will be run on the connection after the service protection criteria have been evaluated. This rule will be executed prior to normal rules configured for the virtual server. Value type: `Reference(config-trafficscript)` Default: `<none>`
`testing`	Place the service protection class into testing mode. (Log when this class would have dropped a connection, but allow all connections through). Value type: `Boolean` Default: `false`

Properties for the `access_restriction` section:	
`allowed`	Always allow access to these IP addresses. This overrides the connection limits for these machines, but does not stop other restrictions such as HTTP validity checks. Value type: `Set(String)` Default: `<none>`
`banned`	Disallow access to these IP addresses. Value type: `Set(String)` Default: `<none>`
Properties for the `connection_limiting` section:	
`max_10_connections`	Maximum simultaneous connections allowed from the top ten busiest IP addresses. Value type: `UInt` Default: `200`
`max_1_connections`	Maximum simultaneous connections allowed from one IP address. Value type: `UInt` Default: `30`
`max_connection_rate`	Maximum number of connections from one IP address in the `rate_timer` interval. Set to `0` to make this unlimited. If applied to an HTTP Virtual Server each request sent on a connection that is kept alive will also be considered. Value type: `UInt` Default: `<none>`
`min_connections`	Always allow at least this number of simultaneous connections from each IP address before applying restrictions. Set to `0` to allow unlimited simultaneous connections. Value type: `UInt` Default: `4`

rate_timer	How frequently the max_connection_rate is assessed. For example, a value of 1 (second) will impose a limit of max_connection_rate connections per *second*; a value of 60 will impose a limit of max_connection_rate connections per *minute*. The valid range is 1-99999 seconds. Value type: UInt Default: 60
Properties for the http section:	
check_rfc2396	Whether or not requests with poorly-formed URLs be should be rejected. This tests URL compliance as defined in RFC2396. Note that enabling this may block some older, non-conforming web browsers. Value type: Boolean Default: false
max_body_length	Maximum permitted length of HTTP request body data, set to 0 to disable the limit. Value type: UInt Default: <none>
max_header_length	Maximum permitted length of a single HTTP request header (key and value), set to 0 to disable the limit. Value type: UInt Default: <none>
max_request_length	Maximum permitted size of all the HTTP request headers, set to 0 to disable the limit. Value type: UInt Default: <none>
max_url_length	Maximum permitted URL length, set to 0 to disable the limit. Value type: UInt Default: <none>

reject_binary	Whether or not URLs and HTTP request headers that contain binary data (after decoding) should be rejected. Value type: Boolean Default: false
send_error_page	This setting tells the traffic manager to send an HTTP error message if a connection fails the service protection tests, instead of just dropping it. Details of which HTTP response will be sent when particular tests fail can be found in the Help section for this page. Value type: Boolean Default: true

Rate Shaping Class

URI Path: rate

A rate shaping class restricts the number of connections being processed by a virtual server at once.

Property	Description
max_rate_per_minute	Requests that are associated with this rate class will be rate-shaped to this many requests per minute, set to 0 to disable the limit. Value type: UInt Default: <none>
max_rate_per_second	Although requests will be rate-shaped to the max_rate_per_minute, the traffic manager will also rate limit per-second. This smooths traffic so that a full minute's traffic will not be serviced in the first second of the minute, set this to 0 to disable the per-second limit. Value type: UInt Default: <none>
note	A description of the rate class. Value type: FreeformString

	Default: <none>

Rule

URI Path: rules

TrafficScript rules allow traffic inspection and modification.

Property	Description
There are no properties to display for this resource.	

SLM Class

URI Path: service_level_monitors

Service level monitoring is used to produce alerts when an application's performance is degraded. This is done by monitoring the response time of connections to a virtual server.

Property	Description
note	A description for the SLM class. Value type: FreeformString Default: <none>
response_time	Responses that arrive within this time limit, expressed in milliseconds, are treated as conforming. Value type: UInt Default: 1000
serious_threshold	When the percentage of conforming responses drops below this level, a serious error level message will be emitted. Value type: UInt Default: <none>
warning_threshold	When the percentage of conforming responses drops below this

	level, a warning message will be emitted. Value type: `UInt` Default: `50`

SSL Client Key Pair

URI Path: `ssl/client_keys`

SSL Client Certificates are used when connecting to backend nodes that require client certificate authentication.

Property	Description
`note`	Notes for this certificate Value type: `FreeformString` Default: `<none>`
`private`	Private key for certificate Value type: `FreeformString` Default: `<none>`
`public`	Public certificate Value type: `FreeformString` Default: `<none>`
`request`	Certificate Signing Request for certificate Value type: `FreeformString` Default: `<none>`

SSL Key Pair

URI Path: `ssl/server_keys`

SSL Server Certificates are presented to clients by virtual servers when SSL decryption is enabled.

Property	Description
`note`	Notes for this certificate Value type: `FreeformString` Default: `<none>`
`private`	Private key for certificate Value type: `FreeformString` Default: `<none>`
`public`	Public certificate Value type: `FreeformString` Default: `<none>`
`request`	Certificate Signing Request for certificate Value type: `FreeformString` Default: `<none>`

SSL Trusted Certificate

URI Path: `ssl/cas`

SSL certificate authority certificates (CAs) and certificate revocation lists (CRLs) can be used when validating server and client certificates.

Property	Description
There are no properties to display for this resource.	

Security Settings

URI Path: `security`

Security settings that restrict remote administration for the cluster. Additional security options can be found in Global Settings.

Property	Description
`access`	Access to the admin server and REST API is restricted by usernames and passwords. You can further restrict access to just trusted IP addresses, CIDR IP subnets or DNS wildcards. These access restrictions are also used when another traffic manager initially joins the cluster, after joining the cluster these restrictions are no longer used. Care must be taken when changing this setting, as it can cause the administration server to become inaccessable.Access to the admin UI will not be affected until it is restarted. Value type: `Set(String)` Default: `<none>`

Session Persistence Class

URI Path: `persistence`

A session persistence class is used to identify the session a new connection belongs too and deliver it to the same backend node.

Property	Description
`cookie`	The cookie name to use for tracking session persistence. Value type: `String` Default: `<none>`
`delete`	Whether or not the session should be deleted when a session failure occurs. (Note, setting a failure mode of 'choose a new node' implicitly deletes the session.)

<table>
<tr><td></td><td>Value type: <code>Boolean</code>

Default: <code>true</code></td></tr>
<tr><td><code>failure_mode</code></td><td>The action the pool should take if the session data is invalid or it cannot contact the node specified by the session.

Value type: <code>Enum(String)</code>

Default: <code>new_node</code>

Permitted values:<table><tr><td><code>close</code></td><td>Close the connection (using the Virtual Servers error file)</td></tr><tr><td><code>new_node</code></td><td>Choose a new node to use</td></tr><tr><td><code>url</code></td><td>Redirect the user to a given URL</td></tr></table></td></tr>
<tr><td><code>note</code></td><td>A description of the session persistence class.

Value type: <code>FreeformString</code>

Default: <code><none></code></td></tr>
<tr><td><code>type</code></td><td>The type of session persistence to use.

Value type: <code>Enum(String)</code>

Default: <code>ip</code>

Permitted values:<table><tr><td><code>asp</code></td><td>ASP and ASP.NET session persistence</td></tr><tr><td><code>cookie</code></td><td>Monitor application cookies</td></tr><tr><td><code>ip</code></td><td>IP-based persistence</td></tr><tr><td><code>j2ee</code></td><td>J2EE session persistence</td></tr><tr><td><code>named</code></td><td>Named Node session persistence</td></tr><tr><td><code>ssl</code></td><td>SSL Session ID persistence</td></tr><tr><td><code>transparent</code></td><td>Transparent session affinity</td></tr><tr><td><code>universal</code></td><td>Universal session persistence</td></tr><tr><td><code>x_zeus</code></td><td>X-Zeus-Backend cookies</td></tr></table></td></tr>
<tr><td><code>url</code></td><td>The redirect URL to send clients to if the session persistence is configured to redirect users when a node dies.</td></tr>
</table>

	Value type: `String` Default: `<none>`

Traffic IP Group

URI Path: `traffic_ip_groups`

Traffic IP groups are sets of IP addresses that are distributed across a cluster for fault tolerance.

Property	Description
`enabled`	If set to `No`, the traffic IP group will be disabled and none of the traffic IP addresses will be raised. Value type: `Boolean` Default: `true`
`hash_source_port`	Whether or not the source port should be taken into account when deciding which traffic manager should handle a request. Value type: `Boolean` Default: `false`
`ip_mapping`	A table assigning traffic IP addresses to machines that should host them. Traffic IP addresses not specified in this table will automatically be assigned to a machine. primary key: `ip (String)` — A traffic IP address (from the ipaddresses property). sub keys: `traffic_manager (String)` — The name of the traffic manager that should host the IP address.
`ipaddresses`	The IP addresses that belong to the Traffic IP group. Value type: `Set(String)` Default: `<none>`
`keeptogether`	If set to `Yes` then all the traffic IPs will be raised on a single traffic manager. By default they're distributed across all active traffic

<table>
<tr><td></td><td>managers in the traffic IP group.

Value type: <code>Boolean</code>

Default: <code>false</code></td></tr>
<tr><td><code>location</code></td><td>The location in which the Traffic IP group is based.

Value type: <code>Int</code>

Default: <code><none></code></td></tr>
<tr><td><code>machines</code></td><td>The traffic managers that can host the traffic IP group's IP addresses.

Value type: <code>Set(Reference(config-tm))</code>

Default: <code><none></code></td></tr>
<tr><td><code>mode</code></td><td>The method used to distribute traffic IPs across machines in the cluster. If "multihosted" is used then <code>multicast</code> must be set to an appropriate multicast IP address.

Value type: <code>Enum(String)</code>

Default: <code>singlehosted</code>

Permitted values:
<table>
<tr><td><code>ec2elastic</code></td><td>Use an EC2-Classic Elastic IP address.</td></tr>
<tr><td><code>ec2vpcelastic</code></td><td>Use an EC2-VPC Elastic IP address.</td></tr>
<tr><td><code>multihosted</code></td><td>Raise each address on every machine in the group (Multi-Hosted mode) - IPv4 only</td></tr>
<tr><td><code>singlehosted</code></td><td>Raise each address on a single machine (Single-Hosted mode)</td></tr>
</table></td></tr>
<tr><td><code>multicast</code></td><td>The multicast IP address used to duplicate traffic to all traffic managers in the group.

Value type: <code>String</code>

Default: <code><none></code></td></tr>
<tr><td><code>note</code></td><td>A description of this traffic IP group.

Value type: <code>String</code></td></tr>
</table>

	Default: `<none>`
`slaves`	A list of traffic managers that are in 'passive' mode. This means that in a fully working environment, they will not have any traffic IP addresses assigned to them. Value type: `Set(Reference(config-tm))` Default: `<none>`

Traffic Manager

URI Path: `traffic_managers`

Settings that alter the behavior of services running on a single machine.

Property	Description
`appliance_sysctl`	Custom kernel parameters applied by the user with sysctl interface primary key: `sysctl (String)` — The name of the kernel parameter, e.g. net.ipv4.forward sub keys: `description (String)` — Associated optional description for the sysctl `value (String)` — The value of the kernel parameter
`location`	This is the location of the local traffic manager is in. Value type: `String` Default: `<none>`
`nameip`	Replace Traffic Manager name with an IP address. Value type: `String` Default: `<none>`
`num_aptimizer_threads`	How many worker threads the Aptimizer process should create to optimise content. By default, one thread will be created for each CPU on the system.

<table>
<tr><td></td><td>Value type: UInt

Default: <none></td></tr>
<tr><td>num_children</td><td>The number of worker processes the software will run. By default, one child process will be created for each CPU on the system. You may wish to reduce this to effectively "reserve" CPU(s) for other processes running on the host system.

Value type: UInt

Default: <none></td></tr>
<tr><td>trafficip</td><td>A table mapping interfaces to networks, used by the traffic manager to select which interface to raise a Traffic IP on.

primary key:<table><tr><td>name (String)</td><td>A network interface.</td></tr></table>
sub keys:<table><tr><td>networks (Set(String))</td><td>A set of IP/masks to which the network interface maps.</td></tr></table></td></tr>
<tr><td colspan="2">Properties for the appliance section:</td></tr>
<tr><td>gateway_ipv4</td><td>The default gateway.

Value type: String

Default: <none></td></tr>
<tr><td>gateway_ipv6</td><td>The default IPv6 gateway.

Value type: String

Default: <none></td></tr>
<tr><td>hostname</td><td>Name (hostname.domainname) of the appliance.

Value type: String

Default: <none></td></tr>
<tr><td>hosts</td><td>A table of hostname to static ip address mappings, to be placed in the /etc/hosts file.

primary key:<table><tr><td>name (String)</td><td>The name of a host.</td></tr></table>
sub keys:<table><tr><td>ip_address</td><td>The static IP address of the host.</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td><table><tr><td>(String)</td><td></td></tr></table></td></tr>
<tr><td>if</td><td>A table of network interface specific settings.

primary key:
<table><tr><td>name (String)</td><td>A network interface name.</td></tr></table>
sub keys:
<table>
<tr><td>autoneg (Boolean)</td><td>Whether auto-negotiation should be enabled for the interface.</td></tr>
<tr><td>bmode (Enum(String))</td><td>The trunking mode used for the interface (only 802.3ad is currently supported).

Permitted values:<table><tr><td>802_3ad</td><td>IEEE 802.3ad</td></tr><tr><td>balance_alb</td><td>Adaptive Load Balancing</td></tr></table></td></tr>
<tr><td>bond (String)</td><td>The trunk of which the interface should be a member.</td></tr>
<tr><td>duplex (Boolean)</td><td>Whether full-duplex should be enabled for the interface.</td></tr>
<tr><td>mtu (UInt)</td><td>The maximum transmission unit (MTU) of the interface.</td></tr>
<tr><td>speed (Enum(String))</td><td>The speed of the interface.

Permitted values:<table><tr><td>10</td><td>10Mbs</td></tr><tr><td>100</td><td>100Mbs</td></tr><tr><td>1000</td><td>1Gbs</td></tr></table></td></tr>
</table></td></tr>
<tr><td>ip</td><td>A table of network interfaces and their network settings.

primary key:
<table><tr><td>name (String)</td><td>A network interface name.</td></tr></table>
sub keys:
<table>
<tr><td>addr (String)</td><td>The IP address for the interface.</td></tr>
<tr><td>isexternal (Boolean)</td><td>Whether the interface is externally facing.</td></tr>
</table></td></tr>
</table>

	`mask` (String) — The IP mask (netmask) for the interface.
`licence_agreed`	Whether or not the license agreement has been accepted. This determines whether or not the Initial Configuration wizard is displayed. Value type: `Boolean` Default: `false`
`manageec2conf`	Whether or not the software manages the EC2 config. Value type: `Boolean` Default: `true`
`manageiptrans`	Whether or not the software manages the IP transparency Value type: `Boolean` Default: `true`
`managereturnpath`	Whether or not the software manages return path routing. If disabled, the appliance won't modify iptables / rules / routes for this feature. Value type: `Boolean` Default: `true`
`managesysctl`	Whether or not the software manages user specified sysctl keys. Value type: `Boolean` Default: `true`
`managevpcconf`	Whether or not the software manages the EC2-VPC secondary IPs. Value type: `Boolean` Default: `true`
`name_servers`	The IP addresses of the nameservers the appliance should use and place in `/etc/resolv.conf`. Value type: `Set(String)`

<table>
<tr><td></td><td>Default: <none></td></tr>
<tr><td>ntpservers</td><td>The NTP servers the appliance should use to synchronize its clock.

Value type: List(String)

Default: 0.riverbed.pool.ntp.org
1.riverbed.pool.ntp.org
2.riverbed.pool.ntp.org
3.riverbed.pool.ntp.org</td></tr>
<tr><td>routes</td><td>A table of destination IP addresses and routing details to reach them.

primary key:<table><tr><td>name (String)</td><td>A destination IP address.</td></tr></table>
sub keys:<table><tr><td>gw (String)</td><td>The gateway IP to configure for the route.</td></tr><tr><td>if (String)</td><td>The network interface to configure for the route.</td></tr><tr><td>mask (String)</td><td>The netmask to apply to the IP address.</td></tr></table></td></tr>
<tr><td>search_domains</td><td>The search domains the appliance should use and place in /etc/resolv.conf.

Value type: Set(String)

Default: <none></td></tr>
<tr><td>shim_client_id</td><td>The client ID provided by the portal for this server.

Value type: String

Default: <none></td></tr>
<tr><td>shim_client_key</td><td>The client key provided by the portal for this server.

Value type: String

Default: <none></td></tr>
<tr><td>shim_enabled</td><td>Enable the Cloud Steelhead discovery agent on this appliance.

Value type: Boolean

Default: false</td></tr>
</table>

<table>
<tr><td>shim_ips</td><td>The IP addresses of the Cloud Steelheads to use, as a space or comma separated list. If using priority load balancing this should be in ascending order of priority (highest priority last).

Value type: String

Default: <none></td></tr>
<tr><td>shim_load_balance</td><td>The load balancing method for the selecting a Cloud Steelhead appliance.

Value type: Enum(String)

Default: round_robin

Permitted values:<table><tr><td>priority</td><td>Priority</td></tr><tr><td>round_robin</td><td>Round Robin</td></tr></table></td></tr>
<tr><td>shim_log_level</td><td>The minimum severity that the discovery agent will record to its log.

Value type: Enum(UInt)

Default: notice

Permitted values:<table><tr><td>critical</td><td>Log critical errors</td></tr><tr><td>debug</td><td>Log debug or more severe errors (all errors)</td></tr><tr><td>info</td><td>Log info or more severe errors</td></tr><tr><td>notice</td><td>Log notice or more severe errors</td></tr><tr><td>serious</td><td>Log serious or more severe errors</td></tr><tr><td>warning</td><td>Log warning or more severe errors</td></tr></table></td></tr>
<tr><td>shim_mode</td><td>The mode used to discover Cloud Steelheads in the local cloud or data center.

Value type: Enum(String)

Default: portal

Permitted values:<table><tr><td>local</td><td>Local Portal</td></tr><tr><td>manual</td><td>Manual</td></tr><tr><td>portal</td><td>Riverbed Portal</td></tr></table></td></tr>
</table>

shim_portal_url	The hostname or IP address of the local portal to use. Value type: String Default: <none>
shim_proxy_host	The IP or hostname of the proxy server to use to connect to the portal. Leave blank to not use a proxy server. Value type: String Default: <none>
shim_proxy_port	The port of the proxy server, must be set if a proxy server has been configured. Value type: String Default: <none>
ssh_enabled	Whether or not the SSH server is enabled on the appliance. Value type: Boolean Default: true
ssh_port	The port that the SSH server should listen on. Value type: UInt Default: 22
timezone	The timezone the appliance should use. This must be a path to a timezone file that exists under /usr/share/zoneinfo/. Value type: String Default: US/Pacific
vlans	The VLANs the software should raise. A VLAN should be configured using the format <dev>.<vlanid>, where <dev> is the name of a network device that exists in the host system, eth0.100 for example. Value type: Set(String) Default: <none>

Properties for the `cluster_comms` section:	
`allow_update`	Whether or not this instance of the software can send configuration updates to other members of the cluster. When not clustered this key is ignored. When clustered the value can only be changed by another machine in the cluster that has `allow_update` set to `true`. If set to `false` then it will not be possible to log into the admin server for this instance. Value type: `Boolean` Default: `true`
`bind_ip`	The IP address that the software should bind to for internal administration communications. See also `port`. If the software is not part of a cluster the default is to use `127.0.0.1` and there should be no reason to touch this setting. If the software is part of a cluster then the default is to listen on all raised IPs, in this case an alternative configuration is to listen on a single IP address. This may be useful if you have a separate management network and wish to restrict control messages to it. It is important to ensure that the `allowed_update_hosts` (in the `Global Settings` resource) is compatible with the IP configured here. Value type: `String` Default: `*`
`external_ip`	This is the optional external ip of the traffic manager, which is used to circumvent natting when traffic managers in a cluster span different networks. Value type: `String` Default: `<none>`
`port`	The port that the software should listen on for internal administration communications. See also `bind_ip`. Value type: `UInt` Default: `9080`
Properties for the `ec2` section:	
`availability_zone`	The availability zone of this EC2 instance, should be set when the appliance is first booted. Not required for non-EC2 systems. Value type: `String`

	Default: <none>
instanceid	The EC2 instance ID of this EC2 virtual appliance, should be set when the appliance is first booted. Not required for non-EC2 systems. Value type: String Default: <none>
vpcid	The ID of the VPC the instance is in, should be set when the appliance is first booted. Not required for non-VPC EC2 or non-EC2 systems. Value type: String Default: <none>
Properties for the java section:	
port	The port the Java Extension handler process should listen on. This port will be bound for localhost communications only. Value type: UInt Default: 9060
Properties for the rest_api section:	
bind_ips	A list of IP Addresses which the REST API will listen on for connections. The list should contain IP addresses (IPv4 or IPv6) or a single entry containing an asterisk (*). This indicates that the REST API should listen on all IP Addresses. Value type: Set(String) Default: *
port	The port on which the REST API should listen for requests. Value type: UInt Default: 9070
Properties for the snmp section:	
allow	Restrict which IP addresses can access the SNMP command responder service. The value can be all, localhost, or a list of IP CIDR subnet masks. For example 10.100.0.0/16 would

<table>
<tr><td></td><td>allow connections from any IP address beginning with <code>10.100</code>.

Value type: <code>Set(String)</code>

Default: <code>all</code></td></tr>
<tr><td><code>auth_password</code></td><td>The authentication password. Required (minimum length 8 characters) if <code>security_level</code> includes authentication.

Value type: <code>Password</code>

Default: <code><none></code></td></tr>
<tr><td><code>bind_ip</code></td><td>The IP address the SNMP service should bind its listen port to. The value * (asterisk) means SNMP will listen on all IP addresses.

Value type: <code>String</code>

Default: <code>*</code></td></tr>
<tr><td><code>community</code></td><td>The community string required for SNMPv1 and SNMPv2c commands. (If empty, all SNMPv1 and SNMPv2c commands will be rejected).

Value type: <code>String</code>

Default: <code>public</code></td></tr>
<tr><td><code>enabled</code></td><td>Whether or not the SNMP command responder service should be enabled on this traffic manager.

Value type: <code>Boolean</code>

Default: <code>false</code></td></tr>
<tr><td><code>hash_algorithm</code></td><td>The hash algorithm for authenticated SNMPv3 communications.

Value type: <code>Enum(String)</code>

Default: <code>md5</code>

Permitted values:<table><tr><td><code>md5</code></td><td>MD5</td></tr><tr><td><code>sha1</code></td><td>SHA-1</td></tr></table></td></tr>
<tr><td><code>port</code></td><td>The port the SNMP command responder service should listen on. The value <code>default</code> denotes port 161 if the software is running with root privileges, and 1161 otherwise.</td></tr>
</table>

	Value type: `String` Default: `default`
`priv_password`	The privacy password. Required (minimum length 8 characters) if `security_level` includes privacy (message encryption). Value type: `Password` Default: `<none>`
`security_level`	The security level for SNMPv3 communications. Value type: `Enum(String)` Default: `noauthnopriv` Permitted values: `authnopriv` — Authentication only `authpriv` — Authentication and Privacy `noauthnopriv` — No Authentication, No Privacy
`username`	The username required for SNMPv3 commands. (If empty, all SNMPv3 commands will be rejected). Value type: `String` Default: `<none>`

TrafficScript Authenticator

URI Path: `rule_authenticators`

TrafficScript authenticators define remote authentication services that can be queried via a TrafficScript rule.

Property	Description
`host`	The hostname or IP address of the remote authenticator. Value type: `String` Default: `<none>`

note	A description of the authenticator. Value type: FreeformString Default: <none>
port	The port on which the remote authenticator should be contacted. Value type: UInt Default: 389
Properties for the ldap section:	
attributes	A list of attributes to return from the search. If blank, no attributes will be returned. If set to '*' then all user attributes will be returned. Value type: Set(String) Default: <none>
bind_dn	The distinguished name (DN) of the 'bind' user. The traffic manager will connect to the LDAP server as this user when searching for user records. Value type: String Default: <none>
bind_password	The password for the bind user. Value type: Password Default: <none>
filter	The filter used to locate the LDAP record for the user being authenticated. Any occurrences of '%u' in the filter will be replaced by the name of the user being authenticated. Value type: String Default: <none>
filter_base_dn	The base distinguished name (DN) under which user records are located on the server. Value type: String

	Default: `<none>`
`ssl_cert`	The SSL certificate that the traffic manager should use to validate the remote server. If no certificate is specified then no signature validation will be performed. Value type: `Reference(config-ssl-cacrl)` Default: `<none>`
`ssl_enabled`	Whether or not to enable SSL encryption to the LDAP server. Value type: `Boolean` Default: `false`
`ssl_type`	The type of LDAP SSL encryption to use. Value type: `Enum(String)` Default: `ldaps` Permitted values: `ldaps` — LDAPS; `starttls` — Start TLS

User Authenticator

URI Path: `user_authenticators`

A user authenticator is used to allow access to the UI and REST API by querying a remote authentication service.

Property	Description
`description`	A description of the authenticator. Value type: `String` Default: `<none>`
`enabled`	Whether or not this authenticator is enabled. Value type: `Boolean`

<table>
<tr><td></td><td>Default: <code>false</code></td></tr>
<tr><td><code>type</code></td><td>The type and protocol used by this authentication service.

Value type: <code>Enum(String)</code>

Default: <code><none></code>

Permitted values:
<table>
<tr><td><code>ldap</code></td><td>LDAP</td></tr>
<tr><td><code>radius</code></td><td>RADIUS</td></tr>
<tr><td><code>tacacs_plus</code></td><td>TACACS+</td></tr>
</table>
</td></tr>
<tr><td colspan="2">Properties for the <code>ldap</code> section:</td></tr>
<tr><td><code>base_dn</code></td><td>The base DN (Distinguished Name) under which directory searches will be applied. The entries for your users should all appear under this DN. An example of a typical base DN is: <code>OU=users, DC=mycompany, DC=local</code>

Value type: <code>String</code>

Default: <code><none></code></td></tr>
<tr><td><code>bind_dn</code></td><td>Template to construct the bind DN (Distinguished Name) from the username. The string <code>%u</code> will be replaced by the username. Examples: <code>%u@mycompany.local</code> for Active Directory or <code>cn=%u, dn=mycompany, dn=local</code> for both LDAP and Active Directory.

Value type: <code>String</code>

Default: <code><none></code></td></tr>
<tr><td><code>dn_method</code></td><td>The bind DN (Distinguished Name) for a user can either be searched for in the directory using the base distinguished name and filter values, or it can be constructed from the username.

Value type: <code>Enum(String)</code>

Default: <code>none</code>

Permitted values:
<table>
<tr><td><code>construct</code></td><td>Construct</td></tr>
<tr><td><code>none</code></td><td>No setting configured</td></tr>
<tr><td><code>search</code></td><td>Search</td></tr>
</table>
</td></tr>
<tr><td><code>fallback_group</code></td><td>If the group attribute is not defined, or returns no results for the</td></tr>
</table>

	user logging in, the group named here will be used. If not specified, users will be denied access to the traffic manager if no groups matching a Permission Group can be found for them in the directory. Value type: `String` Default: `<none>`
`filter`	A filter that can be used to extract a unique user record located under the base DN (Distinguished Name). The string `%u` will be replaced by the username. This filter is used to find a user's bind DN when **dn_method** is set to "Search", and to extract group information if the **group filter** is not specified. Examples: `sAMAccountName=%u` for Active Directory, or `uid=%u` for some Unix LDAP schemas. Value type: `String` Default: `<none>`
`group_attribute`	The LDAP attribute that gives a user's group. If there are multiple entries for the attribute all will be extracted and they'll be lexicographically sorted, then the first one to match a Permission Group name will be used. Value type: `String` Default: `<none>`
`group_field`	The sub-field of the group attribute that gives a user's group. For example, if **group_attribute** is `memberOf` and this retrieves values of the form `CN=mygroup, OU=groups, OU=users, DC=mycompany, DC=local` you would set group_field to `CN`. If there are multiple matching fields only the first matching field will be used. Value type: `String` Default: `<none>`
`group_filter`	If the user record returned by **filter** does not contain the required group information you may specify an alternative group search filter here. This will usually be required if you have Unix/POSIX-style user records. If multiple records are returned the list of group names will be extracted from all of them. The string `%u` will be replaced by the username. Example: `(&(memberUid=%u)(objectClass=posixGroup))` Value type: `String`

	Default: <none>
port	The port to connect to the LDAP server on. Value type: UInt Default: 389
search_dn	The bind DN (Distinguished Name) to use when searching the directory for a user's bind DN. You can leave this blank if it is possible to perform the bind DN search using an anonymous bind. Value type: String Default: <none>
search_password	If binding to the LDAP server using search_dn requires a password, enter it here. Value type: Password Default: <none>
server	The IP or hostname of the LDAP server. Value type: String Default: <none>
timeout	Connection timeout in seconds. Value type: UInt Default: 30
Properties for the radius section:	
fallback_group	If no group is found using the vendor and group identifiers, or the group found is not valid, the group specified here will be used. Value type: String Default: <none>
group_attribute	The RADIUS identifier for the attribute that specifies an account's group. May be left blank if **fallback group** is specified. Value type: UInt

	Default: 1
group_vendor	The RADIUS identifier for the vendor of the RADIUS attribute that specifies an account's group. Leave blank if using a standard attribute (i.e. for Filter-Id set group_attribute to 11). Value type: UInt Default: 7146
nas_identifier	This value is sent to the RADIUS server. Value type: String Default: <none>
nas_ip_address	This value is sent to the RADIUS server, if left blank the address of the interfaced used to connect to the server will be used. Value type: String Default: <none>
port	The port to connect to the RADIUS server on. Value type: UInt Default: 1812
secret	Secret key shared with the RADIUS server. Value type: Password Default: <none>
server	The IP or hostname of the RADIUS server. Value type: String Default: <none>
timeout	Connection timeout in seconds. Value type: UInt Default: 30
Properties for the tacacs_plus section:	

<table>
<tr><td>auth_type</td><td>Authentication type to use.
Value type: Enum(String)
Default: pap
Permitted values:<table><tr><td>ascii</td><td>ASCII</td></tr><tr><td>pap</td><td>PAP</td></tr></table></td></tr>
<tr><td>fallback_group</td><td>If group_service is not used, or no group value is provided for the user by the TACACS+ server, the group specified here will be used. If this is not specified, users with no TACACS+ defined group will be denied access.
Value type: String
Default: <none></td></tr>
<tr><td>group_field</td><td>The TACACS+ "service" field that provides each user's group.
Value type: String
Default: permission-group</td></tr>
<tr><td>group_service</td><td>The TACACS+ "service" that provides each user's group field.
Value type: String
Default: zeus</td></tr>
<tr><td>port</td><td>The port to connect to the TACACS+ server on.
Value type: UInt
Default: 49</td></tr>
<tr><td>secret</td><td>Secret key shared with the TACACS+ server.
Value type: Password
Default: <none></td></tr>
<tr><td>server</td><td>The IP or hostname of the TACACS+ server.
Value type: String
Default: <none></td></tr>
</table>

timeout	Connection timeout in seconds. Value type: UInt Default: 30

User Group

URI Path: user_groups

Permission groups specify permissions for groups of users. These groups can be given read-write or read-only access to different parts of the configuration hierarchy. Each group will contain a table of permissions. Each table entry has a name that corresponds to a part of the configuration hierarchy, and a corresponding access level. The access level may have values of either none, ro (read only, this is the default), or full. Some permissions have sub-permissions, these are denoted by following the parent permission name with a colon (:) followed by the sub-permission name. The built-in admin group has a special permission key of all with the value full, this must not be altered for the admin group but can be used in other group configuration files to change the default permission level for the group.

Property	Description
description	A description for the group. Value type: String Default: <none>
password_expire_time	Members of this group must renew their passwords after this number of days. To disable password expiry for the group set this to 0 (zero). Note that this setting applies only to local users. Value type: UInt Default: <none>
permissions	A table defining which level of permission this group has for specific configuration elements. primary key: name (String) — Configuration element to which this group has a level of permission. sub keys: access_level (String) — Permission level for the configuration element (none,

	ro or full)
`timeout`	Inactive UI sessions will timeout after this number of seconds. To disable inactivity timeouts for the group set this to `0` (zero). Value type: `UInt` Default: `30`

Virtual Server

URI Path: `virtual_servers`

A virtual server represents the front end of a load balanced network service. It processes traffic it receives on a specified port and distributes load over a pool of backend nodes.

Property	Description
`add_cluster_ip`	Whether or not the virtual server should add an "X-Cluster-Client-Ip" header to the request that contains the remote client's IP address. Value type: `Boolean` Default: `true`
`add_x_forwarded_for`	Whether or not the virtual server should append the remote client's IP address to the X-Forwarded-For header. If the header does not exist, it will be added. Value type: `Boolean` Default: `false`
`add_x_forwarded_proto`	Whether or not the virtual server should add an "X-Forwarded-Proto" header to the request that contains the original protocol used by the client to connect to the traffic manager. Value type: `Boolean` Default: `false`

bandwidth_class	The bandwidth management class should this server should use, if any. Value type: Reference(config-bandwidth) Default: <none>
connect_timeout	The time, in seconds, to wait for data from a new connection. If no data is received within this time, the connection will be closed. A value of 0 (zero) will disable the timeout. Value type: UInt Default: 10
enabled	Whether the virtual server is enabled. Value type: Boolean Default: false
ftp_force_server_secure	Whether or not the virtual server should require that incoming FTP data connections from the nodes originate from the same IP address as the node. Value type: Boolean Default: true
glb_services	The associated GLB services for this DNS virtual server. Value type: Set(String) Default: <none>
listen_on_any	Whether to listen on all IP addresses Value type: Boolean Default: true
listen_on_hosts	Hostnames and IP addresses to listen on

	Value type: `Set(String)` Default: `<none>`
`listen_on_traffic_ips`	Traffic IP Groups to listen on Value type: `Set(String)` Default: `<none>`
`note`	A description for the virtual server. Value type: `FreeformString` Default: `<none>`
`pool`	The default pool to use for traffic. Value type: `Reference(config-pool)` Default: `<none>`
`port`	The port on which to listen for incoming connections. Value type: `UInt` Default: `<none>`
`protection_class`	The service protection class that should be used to protect this server, if any. Value type: `String` Default: `<none>`
`protocol`	The protocol that the virtual server is using. Value type: `Enum(String)` Default: `http`

<table>
<tr><td></td><td>Permitted values:<table>
<tr><td><code>client_first</code></td><td>Generic client first</td></tr>
<tr><td><code>dns</code></td><td>DNS (UDP)</td></tr>
<tr><td><code>dns_tcp</code></td><td>DNS (TCP)</td></tr>
<tr><td><code>ftp</code></td><td>FTP</td></tr>
<tr><td><code>http</code></td><td>HTTP</td></tr>
<tr><td><code>https</code></td><td>SSL (HTTPS)</td></tr>
<tr><td><code>imaps</code></td><td>SSL (IMAPS)</td></tr>
<tr><td><code>imapv2</code></td><td>IMAPv2</td></tr>
<tr><td><code>imapv3</code></td><td>IMAPv3</td></tr>
<tr><td><code>imapv4</code></td><td>IMAPv4</td></tr>
<tr><td><code>ldap</code></td><td>LDAP</td></tr>
<tr><td><code>ldaps</code></td><td>SSL (LDAPS)</td></tr>
<tr><td><code>pop3</code></td><td>POP3</td></tr>
<tr><td><code>pop3s</code></td><td>SSL (POP3S)</td></tr>
<tr><td><code>rtsp</code></td><td>RTSP</td></tr>
<tr><td><code>server_first</code></td><td>Generic server first</td></tr>
<tr><td><code>siptcp</code></td><td>SIP (TCP)</td></tr>
<tr><td><code>sipudp</code></td><td>SIP (UDP)</td></tr>
<tr><td><code>smtp</code></td><td>SMTP</td></tr>
<tr><td><code>ssl</code></td><td>SSL</td></tr>
<tr><td><code>stream</code></td><td>Generic streaming</td></tr>
<tr><td><code>telnet</code></td><td>Telnet</td></tr>
<tr><td><code>udp</code></td><td>UDP</td></tr>
<tr><td><code>udpstreaming</code></td><td>UDP - Streaming</td></tr>
</table></td></tr>
<tr><td><code>request_rules</code></td><td>Rules to be applied to incoming requests, in order, comma separated.

Value type: <code>List(String)</code>

Default: <code><none></code></td></tr>
</table>

<table>
<tr><td>response_rules</td><td>Rules to be applied to responses, in order, comma separated.

Value type: List(Reference(config-trafficscript))

Default: <none></td></tr>
<tr><td>slm_class</td><td>The service level monitoring class that this server should use, if any.

Value type: Reference(config-slm)

Default: <none></td></tr>
<tr><td>so_nagle</td><td>Whether or not Nagle's algorithm should be used for TCP connections.

Value type: Boolean

Default: false</td></tr>
<tr><td>ssl_client_cert_headers</td><td>What HTTP headers the virtual server should add to each request to show the data in the client certificate.

Value type: Enum(String)

Default: none

Permitted values:<table><tr><td>all</td><td>Certificate fields and certificate text</td></tr><tr><td>none</td><td>No data</td></tr><tr><td>simple</td><td>Certificate fields</td></tr></table></td></tr>
<tr><td>ssl_decrypt</td><td>Whether or not the virtual server should decrypt incoming SSL traffic.

Value type: Boolean

Default: false</td></tr>
<tr><td colspan="2">Properties for the aptimizer section:</td></tr>
<tr><td>enabled</td><td>Whether the virtual server should aptimize web content.</td></tr>
</table>

	Value type: `Boolean` Default: `false`
`profile`	A table of Aptimizer profiles and the application scopes that apply to them. primary key: `name (String)` — The name of an Aptimizer acceleration profile. sub keys: `urls (Set(String))` — The application scopes which apply to the acceleration profile.
Properties for the `connection` section:	
`keepalive`	Whether or not the virtual server should use keepalive connections with the remote clients. Value type: `Boolean` Default: `true`
`keepalive_timeout`	The length of time that the virtual server should keep an idle keepalive connection before discarding it. A value of `0` (zero) will mean that the keepalives are never closed by the traffic manager. Value type: `UInt` Default: `10`
`max_client_buffer`	The amount of memory, in bytes, that the virtual server should use to store data sent by the client. Larger values will use more memory, but will minimise the number of `read()` and `write()` system calls that the traffic manager must perform. Value type: `UInt` Default: `65536`
`max_server_buffer`	The amount of memory, in bytes, that the virtual server should use to store data returned by the server. Larger values will use more memory, but will minimise the number

	of `read()` and `write()` system calls that the traffic manager must perform. Value type: `UInt` Default: `65536`
`server_first_banner`	If specified, the traffic manager will use the value as the banner to send for server-first protocols such as POP, SMTP and IMAP. This allows rules to use the first part of the client data (such as the username) to select a pool. Value type: `String` Default: `<none>`
`timeout`	A connection should be closed if no additional data has been received for this period of time. A value of `0` (zero) will disable this timeout. Value type: `UInt` Default: `300`
Properties for the `connection_errors` section:	
`error_file`	The error message to be sent to the client when the traffic manager detects an internal or backend error for the virtual server. Value type: `Reference(config-extra-file)` Default: `Default`
Properties for the `cookie` section:	
`domain`	The way in which the traffic manager should rewrite the domain portion of any cookies set by a back-end web server. Value type: `Enum(UInt)` Default: `no_rewrite` Permitted values: `no_rewrite` — Do not rewrite the

<table>
<tr><td></td><td><table>
<tr><td></td><td>domain</td></tr>
<tr><td>set_to_named</td><td>Rewrite the domain to the named domain value</td></tr>
<tr><td>set_to_request</td><td>Rewrite the domain to the host header of the request</td></tr>
</table></td></tr>
<tr><td>new_domain</td><td>The domain to use when rewriting a cookie's domain to a named value.

Value type: String

Default: <none></td></tr>
<tr><td>path_regex</td><td>If you wish to rewrite the path portion of any cookies set by a back-end web server, provide a regular expression to match the path:

Value type: String

Default: <none></td></tr>
<tr><td>path_replace</td><td>If cookie path regular expression matches, it will be replaced by this substitution. Parameters $1-$9 can be used to represent bracketed parts of the regular expression.

Value type: String

Default: <none></td></tr>
<tr><td>secure</td><td>Whether or not the traffic manager should modify the "secure" tag of any cookies set by a back-end web server.

Value type: Enum(UInt)

Default: no_modify

Permitted values:<table>
<tr><td>no_modify</td><td>Do not modify the 'secure' tag</td></tr>
<tr><td>set_secure</td><td>Set the 'secure' tag</td></tr>
<tr><td>unset_secure</td><td>Unset the 'secure' tag</td></tr>
</table></td></tr>
</table>

Properties for the `ftp` section:	
`data_source_port`	The source port to be used for active-mode FTP data connections. If 0, a random high port will be used, otherwise the specified port will be used. If a port below 1024 is required you must first explicitly permit use of low ports with the `data_bind_low` global setting. Value type: `UInt` Default: `<none>`
`force_client_secure`	Whether or not the virtual server should require that incoming FTP data connections from the client originate from the same IP address as the corresponding client control connection. Value type: `Boolean` Default: `true`
`port_range_high`	If non-zero, then this controls the upper bound of the port range to use for FTP data connections. Value type: `UInt` Default: `<none>`
`port_range_low`	If non-zero, then this controls the lower bound of the port range to use for FTP data connections. Value type: `UInt` Default: `<none>`
`ssl_data`	Use SSL on the data connection as well as the control connection (if not enabled it is left to the client and server to negotiate this). Value type: `Boolean` Default: `true`
Properties for the `gzip` section:	

compress_level	Compression level (1-9, 1=low, 9=high). Value type: UInt Default: 1
enabled	Compress web pages sent back by the server. Value type: Boolean Default: false
include_mime	MIME types to compress. Complete MIME types can be used, or a type can end in a '*' to match multiple types. Value type: Set(String) Default: text/html text/plain
max_size	Maximum document size to compress (0 means unlimited). Value type: UInt Default: 10000000
min_size	Minimum document size to compress. Value type: UInt Default: 1000
no_size	Compress documents with no given size. Value type: Boolean Default: true
Properties for the http section:	
chunk_overhead_forwarding	Handling of HTTP chunk overhead. When Stingray receives data from a server or client that consists purely of protocol overhead (contains no payload), forwarding of such segments is delayed until useful payload data arrives

<table>
<tr><td></td><td>(setting "lazy"). Changing this key to "eager" will make Stingray incur the overhead of immediately passing such data on; it should only be used with HTTP peers whose chunk handling requires it.

Value type: <code>Enum(String)</code>

Default: <code>lazy</code>

Permitted values:<table><tr><td><code>eager</code></td><td>Forward all data, even when no new payload information is available.</td></tr><tr><td><code>lazy</code></td><td>Only forward segments when useful payload data is available.</td></tr></table></td></tr>
<tr><td><code>location_regex</code></td><td>If the 'Location' header matches this regular expression, rewrite the header using the 'location_replace' pattern.

Value type: <code>String</code>

Default: <code><none></code></td></tr>
<tr><td><code>location_replace</code></td><td>If the 'Location' header matches the 'location_regex' regular expression, rewrite the header with this pattern (parameters such as $1-$9 can be used to match parts of the regular expression):

Value type: <code>String</code>

Default: <code><none></code></td></tr>
<tr><td><code>location_rewrite</code></td><td>The action the virtual server should take if the "Location" header does not match the <code>location_regex</code> regular expression.

Value type: <code>Enum(UInt)</code>

Default: <code>if_host_matches</code>

Permitted values:<table><tr><td><code>always</code></td><td>Rewrite the hostname to the request's "Host" header, and rewrite the protocol and port if necessary;</td></tr></table></td></tr>
</table>

	<table><tr><td>if_host_matches</td><td>Do not rewrite the hostname. Rewrite the protocol and port if the hostname matches the request's "Host" header.</td></tr><tr><td>never</td><td>Nothing;</td></tr></table>
mime_default	Auto-correct MIME types if the server sends the "default" MIME type for files. Value type: String Default: text/plain
mime_detect	Auto-detect MIME types if the server does not provide them. Value type: Boolean Default: false
Properties for the log section:	
client_connection_failures	Should the virtual server log failures occurring on connections to clients. Value type: Boolean Default: false
enabled	Whether or not to log connections to the virtual server to a disk on the file system. Value type: Boolean Default: false
filename	The name of the file in which to store the request logs. Appliances will ignore this. The filename can contain macros which will be expanded by the traffic manager to generate the full filename. Value type: String

	Default: `%zeushome%/zxtm/log/%v.log`
`format`	The log file format. This specifies the line of text that will be written to the log file when a connection to the traffic manager is completed. Many parameters from the connection can be recorded using macros. Value type: `String` Default: `%h %l %u %t "%r" %s %b "%{Referer}i" "%{User-agent}i"`
`server_connection_failures`	Should the virtual server log failures occurring on connections to nodes. Value type: `Boolean` Default: `false`
`ssl_failures`	Should the virtual server log failures occurring on SSL secure negotiation. Value type: `Boolean` Default: `false`
Properties for the `request_tracing` section:	
`enabled`	Record a trace of major connection processing events for each request and response. Value type: `Boolean` Default: `false`
`trace_io`	Include details of individual I/O events in request and response traces. Requires request tracing to be enabled. Value type: `Boolean` Default: `false`
Properties for the `rtsp` section:	

<table>
<tr><td>streaming_port_range_high</td><td>If non-zero this controls the upper bound of the port range to use for streaming data connections.

Value type: UInt

Default: <none></td></tr>
<tr><td>streaming_port_range_low</td><td>If non-zero this controls the lower bound of the port range to use for streaming data connections.

Value type: UInt

Default: <none></td></tr>
<tr><td>streaming_timeout</td><td>If non-zero data-streams associated with RTSP connections will timeout if no data is transmitted for this many seconds.

Value type: UInt

Default: 30</td></tr>
<tr><td colspan="2">Properties for the sip section:</td></tr>
<tr><td>dangerous_requests</td><td>The action to take when a SIP request with body data arrives that should be routed to an external IP.

Value type: Enum(String)

Default: node

Permitted values:<table><tr><td>forbid</td><td>Send a 403 Forbidden response to the client</td></tr><tr><td>forward</td><td>Forward the request to its target URI (dangerous)</td></tr><tr><td>node</td><td>Send the request to a back-end node</td></tr></table></td></tr>
<tr><td>follow_route</td><td>Should the virtual server follow routing information contained in SIP requests. If set to No requests will be routed to the chosen back-end node regardless of their URI or Route header.

Value Boolean</td></tr>
</table>

<table>
<tr><td></td><td>type:

Default: <code>true</code></td></tr>
<tr><td><code>max_connection_mem</code></td><td>SIP clients can have several pending requests at one time. To protect the traffic manager against DoS attacks, this setting limits the amount of memory each client can use. When the limit is reached new requests will be sent a 413 response. If the value is set to <code>0</code> (zero) the memory limit is disabled.

Value type: <code>UInt</code>

Default: <code>65536</code></td></tr>
<tr><td><code>mode</code></td><td>The mode that this SIP virtual server should operate in.

Value type: <code>Enum(String)</code>

Default: <code>sip_gateway</code>

Permitted values:<table><tr><td><code>full_gateway</code></td><td>All SIP requests and responses and all session data will pass through Stingray. A port range to use for the session data and a timeout value for inactive data connections can be specified in the additional settings that are displayed when the <code>Full Gateway</code> mode is selected.</td></tr><tr><td><code>route</code></td><td>The first SIP request in a session will pass through Stingray, along with its responses, but all future requests that are part of the same session will go directly to the back-end node that was chosen by the traffic manager.</td></tr><tr><td><code>sip_gateway</code></td><td>All SIP requests and responses will pass through the traffic manager.</td></tr></table></td></tr>
<tr><td><code>rewrite_uri</code></td><td>Replace the Request-URI of SIP requests with the address of the selected back-end node.</td></tr>
</table>

	Value type: `Boolean` Default: `false`
`streaming_port_range_high`	If non-zero this controls the upper bound of the port range to use for streaming data connections. Value type: `UInt` Default: `<none>`
`streaming_port_range_low`	If non-zero, then this controls the lower bound of the port range to use for streaming data connections. Value type: `UInt` Default: `<none>`
`streaming_timeout`	If non-zero a UDP stream will timeout when no data has been seen within this time. Value type: `UInt` Default: `60`
`timeout_messages`	When timing out a SIP transaction, send a 'timed out' response to the client and, in the case of an INVITE transaction, a CANCEL request to the server. Value type: `Boolean` Default: `true`
`transaction_timeout`	The virtual server should discard a SIP transaction when no further messages have been seen within this time. Value type: `UInt` Default: `30`
Properties for the `smtp` section:	

`expect_starttls`	Whether or not the traffic manager should expect the connection to start off in plain text and then upgrade to SSL using STARTTLS when handling SMTP traffic. Value type: `Boolean` Default: `true`
Properties for the `ssl` section:	
`add_http_headers`	Whether or not the virtual server should add HTTP headers to each request to show the SSL connection parameters. Value type: `Boolean` Default: `false`
`client_cert_cas`	The certificate authorities that this virtual server should trust to validate client certificates. If no certificate authorities are selected, and client certificates are requested, then all client certificates will be accepted. Value type: `Set(String)` Default: `<none>`
`issued_certs_never_expire`	When the virtual server verifies certificates signed by these certificate authorities, it doesn't check the 'not after' date, i.e., they are considered valid even after their expiration date has passed (but not if they have been revoked). Value type: `Set(String)` Default: `<none>`
`ocsp_enable`	Whether or not the traffic manager should use OCSP to check the revocation status of client certificates. Value type: `Boolean` Default: `false`
`ocsp_issuers`	A table of certificate issuer specific OCSP settings.

primary key:

`issuer (String)`	The name of an issuer (or DEFAULT for default OCSP settings).

sub keys:

`aia (Boolean)`	Whether the traffic manager should use AIA information contained in a client certificate to determine which OCSP responder to contact.
`nonce (Enum(String))`	How to use the OCSP nonce extension, which protects against OCSP replay attacks. Some OCSP servers do not support nonces. Permitted values: `off` — No nonce check `on` — Use nonce, server does not have to reply with nonce `strict` — Use nonce, server must reply with nonce
`required (Enum(String))`	Whether we should do an OCSP check for this issuer, and whether it is required or optional. Permitted values: `none` — None `optional` — OCSP check optional `strict` — OCSP check required
`responder_cert (String)`	The expected responder

<table>
<tr><td></td><td><table>
<tr><td></td><td>certificate.</td></tr>
<tr><td>signer (String)</td><td>The certificate with which to sign the request, if any.</td></tr>
<tr><td>url (String)</td><td>Which OCSP responders this virtual server should use to verify client certificates.</td></tr>
</table></td></tr>
<tr><td>ocsp_max_response_age</td><td>The number of seconds for which an OCSP response is considered valid if it has not yet exceeded the time specified in the 'nextUpdate' field. If set to 0 (zero) then OCSP responses are considered valid until the time specified in their 'nextUpdate' field.

Value type: UInt

Default: <none></td></tr>
<tr><td>ocsp_stapling</td><td>If OCSP URIs are present in certificates used by this virtual server, then enabling this option will allow the traffic manager to provide OCSP responses for these certificates as part of the handshake, if the client sends a TLS status_request extension in the ClientHello.

Value type: Boolean

Default: false</td></tr>
<tr><td>ocsp_time_tolerance</td><td>The number of seconds outside the permitted range for which the 'thisUpdate' and 'nextUpdate' fields of an OCSP response are still considered valid.

Value type: UInt

Default: 30</td></tr>
<tr><td>ocsp_timeout</td><td>The number of seconds after which OCSP requests will be timed out.

Value type: UInt

Default: 10</td></tr>
</table>

<table>
<tr><td>prefer_sslv3</td><td>Whether or not to prefer SSLv3 over TLS when the client appears to support both. SSLv3 is slightly faster, but some clients don't allow SSLv3 but still send the ClientHello inside SSLv2 or SSLv3 records. The default option is to prefer TLS due to known vulnerabilities in the way block ciphers are used before TLSv1.1.

Value type: Boolean

Default: false</td></tr>
<tr><td>request_client_cert</td><td>Whether or not the virtual server should request an identifying SSL certificate from each client.

Value type: Enum(UInt)

Default: dont_request

Permitted values:<table><tr><td>dont_request</td><td>Do not request a client certificate</td></tr><tr><td>request</td><td>Request, but do not require a client certificate</td></tr><tr><td>require</td><td>Require a client certificate</td></tr></table></td></tr>
<tr><td>send_close_alerts</td><td>Whether or not to send an SSL/TLS "close alert" when the traffic manager is initiating an SSL socket disconnection.

Value type: Boolean

Default: false</td></tr>
<tr><td>server_cert_default</td><td>The default SSL certificate to use for this virtual server.

Value type: String

Default: <none></td></tr>
<tr><td>server_cert_host_mapping</td><td>Host specific SSL server certificate mappings.

primary key:<table><tr><td>host (String)</td><td>Host which this entry refers to.</td></tr></table>sub keys:<table><tr><td>certificate</td><td>The SSL server certificate for a particular destination</td></tr></table></td></tr>
</table>

	(String) — site IP.
trust_magic	If the traffic manager is receiving traffic sent from another traffic manager, then enabling this option will allow it to decode extra information on the true origin of the SSL connection. This information is supplied by the first traffic manager. Value type: Boolean Default: false
Properties for the syslog section:	
enabled	Whether or not to log connections to the virtual server to a remote syslog host. Value type: Boolean Default: false
format	The log format for the remote syslog. This specifies the line of text that will be sent to the remote syslog when a connection to the traffic manager is completed. Many parameters from the connection can be recorded using macros. Value type: String Default: %h %l %u %t "%r" %s %b "%{Referer}i" "%{User-agent}i"
ip_end_point	The remote host and port (default is 514) to send request log lines to. Value type: String Default: <none>
msg_len_limit	Maximum length in bytes of a message sent to the remote syslog. Messages longer than this will be truncated before they are sent. Value type: UInt

	Default: `1024`
Properties for the `tcp` section:	
`proxy_close`	If set to `Yes` the traffic manager will send the client FIN to the back-end server and wait for a server response instead of closing the connection immediately. This is only necessary for protocols that require half-close support to function correctly, such as "rsh". If the traffic manager is responding to the request itself, setting this key to Yes will cause the traffic manager to continue writing the response even after it has received a FIN from the client. Value type: `Boolean` Default: `false`
Properties for the `udp` section:	
`end_point_persistence`	Whether or not UDP datagrams from the same IP and port are sent to the same node in the pool if there's an existing UDP transaction. Although it's not always guaranteed as while making a decision to reuse the same node, traffic manager can also apply other protocol specific filtering e.g CallID matching for SIP packets in addition to IP and port matching. Value type: `Boolean` Default: `true`
`port_smp`	Whether or not UDP datagrams should be distributed across all traffic manager processes. This setting is not recommended if the traffic manager will be handling connection-based UDP protocols. Value type: `Boolean` Default: `false`
`response_datagrams_expected`	The virtual server should discard any UDP connection and reclaim resources when the node has responded with this number of datagrams. For simple request/response protocols this can be often set to `1`. If set to `-1`, the connection will not be discarded until the `timeout` is reached.

	Value type: `Int` Default: `1`
`timeout`	The virtual server should discard any UDP connection and reclaim resources when no further UDP traffic has been seen within this time. Value type: `UInt` Default: `7`
Properties for the `web_cache` section:	
`control_out`	The "Cache-Control" header to add to every cached HTTP response, `no-cache` or `max-age=600` for example. Value type: `String` Default: `<none>`
`enabled`	If set to `Yes` the traffic manager will attempt to cache web server responses. Value type: `Boolean` Default: `false`
`error_page_time`	Time period to cache error pages for. Value type: `UInt` Default: `30`
`max_time`	Maximum time period to cache web pages for. Value type: `UInt` Default: `600`
`refresh_time`	If a cached page is about to expire within this time, the traffic manager will start to forward some new requests on to the web servers. A maximum of one request per second

	will be forwarded; the remainder will continue to be served from the cache. This prevents "bursts" of traffic to your web servers when an item expires from the cache. Setting this value to `0` will stop the traffic manager updating the cache before it expires. Value type: `UInt` Default: `2`

SNMP Counter Resources

Actions

URI Path: `statistics/actions/*`

Actions statistics values.

Counter	Description
`processed`	Number of times this action has been processed. Value type: `UInt` SNMP Name: `actionsProcessed`

Asp session cache

URI Path: `statistics/cache/asp_session_cache`

Asp session cache statistics values.

Counter	Description
`entries`	The total number of ASP sessions stored in the cache. Value type: `UInt` SNMP Name: `aspSessionCacheEntries`

entries_max	The maximum number of ASP sessions in the cache. Value type: UInt SNMP Name: aspSessionCacheEntriesMax
hit_rate	The percentage of ASP session lookups that succeeded. Value type: UInt SNMP Name: aspSessionCacheHitRate
hits	Number of times a ASP session entry has been successfully found in the cache. Value type: UInt SNMP Name: aspSessionCacheHits
lookups	Number of times a ASP session entry has been looked up in the cache. Value type: UInt SNMP Name: aspSessionCacheLookups
misses	Number of times a ASP session entry has not been available in the cache. Value type: UInt SNMP Name: aspSessionCacheMisses
oldest	The age of the oldest ASP session in the cache (in seconds). Value type: UInt SNMP Name: aspSessionCacheOldest

Bandwidth

URI Path: `statistics/bandwidth/*`

Bandwidth statistics values.

Counter	Description
`bytes_out`	Bytes output by connections assigned to this bandwidth class. Value type: `UInt64` SNMP Name: `bandwidthClassBytesOut`
`bytes_out_hi`	Bytes output by connections assigned to this bandwidth class (high 32bits). Value type: `UInt` SNMP Name: `bandwidthClassBytesOutHi`
`bytes_out_lo`	Bytes output by connections assigned to this bandwidth class (low 32bits). Value type: `UInt` SNMP Name: `bandwidthClassBytesOutLo`
`guarantee`	Guaranteed bandwidth class limit (kbits/s). Currently unused. Value type: `UInt` SNMP Name: `bandwidthClassGuarantee`
`maximum`	Maximum bandwidth class limit (kbits/s). Value type: `UInt` SNMP Name: `bandwidthClassMaximum`

Cloud api credentials

URI Path: `statistics/cloud_api_credentials/*`

Cloud api credentials statistics values.

Counter	Description
`node_creations`	The number of instance creation API requests made with this set of cloud credentials. Value type: `UInt` SNMP Name: `cloudcredentialsNodeCreations`
`node_deletions`	The number of instance destruction API requests made with this set of cloud credentials. Value type: `UInt` SNMP Name: `cloudcredentialsNodeDeletions`
`status_requests`	The number of status API requests made with this set of cloud credentials. Value type: `UInt` SNMP Name: `cloudcredentialsStatusRequests`

Connection rate limit

URI Path: `statistics/connection_rate_limit/*`

Connection rate limit statistics values.

Counter	Description
`conns_entered`	Connections that have entered the rate class and have been queued. Value type: `UInt` SNMP `rateClassConnsEntered`

	Name:
conns_left	Connections that have left the rate class. Value type: UInt SNMP Name: rateClassConnsLeft
current_rate	The average rate that requests are passing through this rate class. Value type: UInt SNMP Name: rateClassCurrentRate
dropped	Requests dropped from this rate class without being processed (e.g. timeouts). Value type: UInt SNMP Name: rateClassDropped
max_rate_per_min	The maximum rate that requests may pass through this rate class (requests/min). Value type: UInt SNMP Name: rateClassMaxRatePerMin
max_rate_per_sec	The maximum rate that requests may pass through this rate class (requests/sec). Value type: UInt SNMP Name: rateClassMaxRatePerSec
queue_length	The current number of requests queued by this rate class. Value type: UInt SNMP Name: rateClassQueueLength

Events

URI Path: `statistics/events/*`

Events statistics values.

Counter	Description
`matched`	Number of times this event configuration has matched. Value type: `UInt` SNMP Name: `eventsMatched`

Glb services

URI Path: `statistics/glb_services/*`

Glb services statistics values.

Counter	Description
`discarded`	Number of A records this GLB Service has discarded. Value type: `UInt` SNMP Name: `glbServiceDiscarded`
`responses`	Number of A records this GLB Service has altered. Value type: `UInt` SNMP Name: `glbServiceResponses`
`unmodified`	Number of A records this GLB Service has passed through unmodified. Value type: `UInt` SNMP Name: `glbServiceUnmodified`

Globals

URI Path: `statistics/globals`

Globals statistics values.

Counter	Description
`data_entries`	Number of entries in the TrafficScript data.get()/set() storage. Value type: `UInt` SNMP Name: `dataEntries`
`data_memory_usage`	Number of bytes used in the TrafficScript data.get()/set() storage. Value type: `UInt` SNMP Name: `dataMemoryUsage`
`events_seen`	Events seen by the traffic Manager's event handling process. Value type: `UInt` SNMP Name: `eventsSeen`
`hourly_peak_bytes_in_per_second`	The peak bytes received from clients per second in the last hour. Value type: `UInt` SNMP Name: `hourlyPeakBytesInPerSecond`
`hourly_peak_bytes_out_per_second`	The peak bytes sent to clients per second in the

	last hour. Value type: `UInt` SNMP Name: `hourlyPeakBytesOutPerSecond`
`hourly_peak_requests_per_second`	The peak requests per second in the last hour. Value type: `UInt` SNMP Name: `hourlyPeakRequestsPerSecond`
`hourly_peak_ssl_connections_per_second`	The peak ssl connections per second in the last hour. Value type: `UInt` SNMP Name: `hourlyPeakSSLConnectionsPerSecond`
`num_idle_connections`	Total number of idle HTTP connections to all nodes (used for future HTTP requests). Value type: `UInt` SNMP Name: `numIdleConnections`
`number_child_processes`	The number of traffic manager child processes. Value type: `UInt` SNMP Name: `numberChildProcesses`
`number_dnsa_cache_hits`	Requests for DNS A records resolved from the traffic manager's local cache. Value type: `UInt`

	SNMP Name: `numberDNSACacheHits`
`number_dnsa_requests`	Requests for DNS A records (hostname->IP address) made by the traffic manager. Value type: `UInt` SNMP Name: `numberDNSARequests`
`number_dnsptr_cache_hits`	Requests for DNS PTR records resolved from the traffic manager's local cache. Value type: `UInt` SNMP Name: `numberDNSPTRCacheHits`
`number_dnsptr_requests`	Requests for DNS PTR records (IP address->hostname) made by the traffic manager. Value type: `UInt` SNMP Name: `numberDNSPTRRequests`
`number_snmp_bad_requests`	Malformed SNMP requests received. Value type: `UInt` SNMP Name: `numberSNMPBadRequests`
`number_snmp_get_bulk_requests`	SNMP GetBulkRequests received. Value type: `UInt` SNMP Name: `numberSNMPGetBulkRequests`
`number_snmp_get_next_requests`	SNMP GetNextRequests received. Value `UInt`

	type: SNMP Name: `numberSNMPGetNextRequests`
`number_snmp_get_requests`	SNMP GetRequests received. Value type: `UInt` SNMP Name: `numberSNMPGetRequests`
`number_snmp_unauthorised_requests`	SNMP requests dropped due to access restrictions. Value type: `UInt` SNMP Name: `numberSNMPUnauthorisedRequests`
`ssl_cipher_3des_decrypts`	Bytes decrypted with 3DES. Value type: `UInt` SNMP Name: `sslCipher3DESDecrypts`
`ssl_cipher_3des_encrypts`	Bytes encrypted with 3DES. Value type: `UInt` SNMP Name: `sslCipher3DESEncrypts`
`ssl_cipher_aes_decrypts`	Bytes decrypted with AES. Value type: `UInt` SNMP Name: `sslCipherAESDecrypts`
`ssl_cipher_aes_encrypts`	Bytes encrypted with AES. Value `UInt`

	type: SNMP Name: `sslCipherAESEncrypts`
`ssl_cipher_decrypts`	Bytes decrypted with a symmetric cipher. Value type: `UInt` SNMP Name: `sslCipherDecrypts`
`ssl_cipher_des_decrypts`	Bytes decrypted with DES. Value type: `UInt` SNMP Name: `sslCipherDESDecrypts`
`ssl_cipher_des_encrypts`	Bytes encrypted with DES. Value type: `UInt` SNMP Name: `sslCipherDESEncrypts`
`ssl_cipher_encrypts`	Bytes encrypted with a symmetric cipher. Value type: `UInt` SNMP Name: `sslCipherEncrypts`
`ssl_cipher_rc4_decrypts`	Bytes decrypted with RC4. Value type: `UInt` SNMP Name: `sslCipherRC4Decrypts`
`ssl_cipher_rc4_encrypts`	Bytes encrypted with RC4. Value type: `UInt`

	SNMP Name: `sslCipherRC4Encrypts`
`ssl_cipher_rsa_decrypts`	Number of RSA decrypts. Value type: `UInt` SNMP Name: `sslCipherRSADecrypts`
`ssl_cipher_rsa_decrypts_external`	Number of external RSA decrypts. Value type: `UInt` SNMP Name: `sslCipherRSADecryptsExternal`
`ssl_cipher_rsa_encrypts`	Number of RSA encrypts. Value type: `UInt` SNMP Name: `sslCipherRSAEncrypts`
`ssl_cipher_rsa_encrypts_external`	Number of external RSA encrypts. Value type: `UInt` SNMP Name: `sslCipherRSAEncryptsExternal`
`ssl_client_cert_expired`	Number of times a client certificate has expired. Value type: `UInt` SNMP Name: `sslClientCertExpired`
`ssl_client_cert_invalid`	Number of times a client certificate was invalid. Value type: `UInt` SNMP `sslClientCertInvalid`

	Name:
ssl_client_cert_not_sent	Number of times a client certificate was required but not supplied. Value type: UInt SNMP Name: sslClientCertNotSent
ssl_client_cert_revoked	Number of times a client certificate was revoked. Value type: UInt SNMP Name: sslClientCertRevoked
ssl_connections	Number of SSL connections negotiated. Value type: UInt SNMP Name: sslConnections
ssl_handshake_sslv2	Number of SSLv2 handshakes. Value type: UInt SNMP Name: sslHandshakeSSLv2
ssl_handshake_sslv3	Number of SSLv3 handshakes. Value type: UInt SNMP Name: sslHandshakeSSLv3
ssl_handshake_t_l_sv1	Number of TLSv1.0 handshakes. Value type: UInt SNMP sslHandshakeTLSv1

	Name:
ssl_handshake_t_l_sv11	Number of TLSv1.1 handshakes. Value type: UInt SNMP Name: sslHandshakeTLSv11
ssl_session_id_disk_cache_hit	Number of times the SSL session id was found in the disk cache and reused (deprecated, will always return 0). Value type: UInt SNMP Name: sslSessionIDDiskCacheHit
ssl_session_id_disk_cache_miss	Number of times the SSL session id was not found in the disk cache (deprecated, will always return 0). Value type: UInt SNMP Name: sslSessionIDDiskCacheMiss
ssl_session_id_mem_cache_hit	Number of times the SSL session id was found in the cache and reused. Value type: UInt SNMP Name: sslSessionIDMemCacheHit
ssl_session_id_mem_cache_miss	Number of times the SSL session id was not found in the cache. Value type: UInt SNMP Name: sslSessionIDMemCacheMiss

sys_cpu_busy_percent	Percentage of time that the CPUs are busy. Value type: UInt SNMP Name: sysCPUBusyPercent
sys_cpu_idle_percent	Percentage of time that the CPUs are idle. Value type: UInt SNMP Name: sysCPUIdlePercent
sys_cpu_system_busy_percent	Percentage of time that the CPUs are busy running system code. Value type: UInt SNMP Name: sysCPUSystemBusyPercent
sys_cpu_user_busy_percent	Percentage of time that the CPUs are busy running user-space code. Value type: UInt SNMP Name: sysCPUUserBusyPercent
sys_fds_free	Number of free file descriptors. Value type: UInt SNMP Name: sysFDsFree
sys_mem_buffered	Buffer memory (MBytes). Value type: UInt SNMP Name: sysMemBuffered

sys_mem_free	Free memory (MBytes). Value type: UInt SNMP Name: sysMemFree
sys_mem_in_use	Memory used (MBytes). Value type: UInt SNMP Name: sysMemInUse
sys_mem_swap_total	Total swap space (MBytes). Value type: UInt SNMP Name: sysMemSwapTotal
sys_mem_swapped	Amount of swap space in use (MBytes). Value type: UInt SNMP Name: sysMemSwapped
sys_mem_total	Total memory (MBytes). Value type: UInt SNMP Name: sysMemTotal
time_last_config_update	The time (in hundredths of a second) since the configuration of traffic manager was updated (this value will wrap if no configuration changes are made for 497 days). Value type: UInt SNMP timeLastConfigUpdate

	Name:
total_backend_server_errors	Total errors returned from the backend servers. Value type: UInt SNMP Name: totalBackendServerErrors
total_bad_dns_packets	Total number of malformed DNS response packets encountered from the backend servers. Value type: UInt SNMP Name: totalBadDNSPackets
total_bytes_in	Bytes received by the traffic manager from clients. Value type: UInt64 SNMP Name: totalBytesIn
total_bytes_in_hi	Bytes received by the traffic manager from clients (high 32bits). Value type: UInt SNMP Name: totalBytesInHi
total_bytes_in_lo	Bytes received by the traffic manager from clients (low 32bits). Value type: UInt SNMP Name: totalBytesInLo
total_bytes_out	Bytes sent by the traffic manager to clients. Value UInt64

	type: SNMP Name: `totalBytesOut`
`total_bytes_out_hi`	Bytes sent by the traffic manager to clients (high 32bits). Value type: `UInt` SNMP Name: `totalBytesOutHi`
`total_bytes_out_lo`	Bytes sent by the traffic manager to clients (low 32bits). Value type: `UInt` SNMP Name: `totalBytesOutLo`
`total_conn`	Total number of TCP connections received. Value type: `UInt` SNMP Name: `totalConn`
`total_current_conn`	Number of TCP connections currently established. Value type: `UInt` SNMP Name: `totalCurrentConn`
`total_dns_responses`	Total number of DNS response packets handled. Value type: `UInt` SNMP Name: `totalDNSResponses`

total_requests	Total number of TCP requests recieved. Value type: UInt SNMP Name: totalRequests
total_transactions	Total number of TCP requests being processed, after applying TPS limits. Value type: UInt SNMP Name: totalTransactions
up_time	The time (in hundredths of a second) that Stingray software has been operational for (this value will wrap if it has been running for more than 497 days). Value type: UInt SNMP Name: upTime

Ip gateway

URI Path: statistics/traffic_ips/ip_gateway

Ip gateway statistics values.

Counter	Description
arp_message	Number of ARP messages sent for raised Traffic IP Addresses. Value type: UInt SNMP Name: trafficIPARPMessage
gateway_ping_requests	Number of ping requests sent to the gateway machine.

	Value type: `UInt` SNMP Name: `trafficIPGatewayPingRequests`
`gateway_ping_responses`	Number of ping responses received from the gateway machine. Value type: `UInt` SNMP Name: `trafficIPGatewayPingResponses`
`node_ping_requests`	Number of ping requests sent to the backend nodes. Value type: `UInt` SNMP Name: `trafficIPNodePingRequests`
`node_ping_responses`	Number of ping responses received from the backend nodes. Value type: `UInt` SNMP Name: `trafficIPNodePingResponses`
`number`	The number of traffic IPv4 addresses on this system. Value type: `UInt` SNMP Name: `trafficIPNumber`
`number_inet46`	The number of traffic IP addresses on this system (includes IPv4 and IPv6 addresses). Value type: `UInt` SNMP Name: `trafficIPNumberInet46`
`number_raised`	The number of traffic IPv4 addresses currently raised on this system. Value type: `UInt` SNMP Name: `trafficIPNumberRaised`

number_raised_inet46	The number of traffic IP addresses currently raised on this system (includes IPv4 and IPv6 addresses). Value type: UInt SNMP Name: trafficIPNumberRaisedInet46
ping_response_errors	Number of ping response errors. Value type: UInt SNMP Name: trafficIPPingResponseErrors

Ip session cache

URI Path: statistics/cache/ip_session_cache

Ip session cache statistics values.

Counter	Description
entries	The total number of IP sessions stored in the cache. Value type: UInt SNMP Name: ipSessionCacheEntries
entries_max	The maximum number of IP sessions in the cache. Value type: UInt SNMP Name: ipSessionCacheEntriesMax
hit_rate	The percentage of IP session lookups that succeeded. Value type: UInt SNMP Name: ipSessionCacheHitRate
hits	Number of times a IP session entry has been successfully found in

	the cache. Value type: `UInt` SNMP Name: `ipSessionCacheHits`
`lookups`	Number of times a IP session entry has been looked up in the cache. Value type: `UInt` SNMP Name: `ipSessionCacheLookups`
`misses`	Number of times a IP session entry has not been available in the cache. Value type: `UInt` SNMP Name: `ipSessionCacheMisses`
`oldest`	The age of the oldest IP session in the cache (in seconds). Value type: `UInt` SNMP Name: `ipSessionCacheOldest`

J2ee session cache

URI Path: `statistics/cache/j2ee_session_cache`

J2ee session cache statistics values.

Counter	Description
`entries`	The total number of J2EE sessions stored in the cache. Value type: `UInt` SNMP Name: `j2eeSessionCacheEntries`

entries_max	The maximum number of J2EE sessions in the cache. Value type: UInt SNMP Name: j2eeSessionCacheEntriesMax
hit_rate	The percentage of J2EE session lookups that succeeded. Value type: UInt SNMP Name: j2eeSessionCacheHitRate
hits	Number of times a J2EE session entry has been successfully found in the cache. Value type: UInt SNMP Name: j2eeSessionCacheHits
lookups	Number of times a J2EE session entry has been looked up in the cache. Value type: UInt SNMP Name: j2eeSessionCacheLookups
misses	Number of times a J2EE session entry has not been available in the cache. Value type: UInt SNMP Name: j2eeSessionCacheMisses
oldest	The age of the oldest J2EE session in the cache (in seconds). Value type: UInt SNMP Name: j2eeSessionCacheOldest

Listen ips

URI Path: `statistics/listen_ips/*`

Listen ips statistics values.

Counter	Description
`bytes_in`	Bytes sent to this listening IP. Value type: `UInt64` SNMP Name: `listenIPBytesIn`
`bytes_in_hi`	Bytes sent to this listening IP (high 32bits). Value type: `UInt` SNMP Name: `listenIPBytesInHi`
`bytes_in_lo`	Bytes sent to this listening IP (low 32bits). Value type: `UInt` SNMP Name: `listenIPBytesInLo`
`bytes_out`	Bytes sent from this listening IP. Value type: `UInt64` SNMP Name: `listenIPBytesOut`
`bytes_out_hi`	Bytes sent from this listening IP (high 32bits). Value type: `UInt` SNMP Name: `listenIPBytesOutHi`
`bytes_out_lo`	Bytes sent from this listening IP (low 32bits). Value type: `UInt` SNMP Name: `listenIPBytesOutLo`

current_conn	TCP connections currently established to this listening IP. Value type: UInt SNMP Name: listenIPCurrentConn
max_conn	Maximum number of simultaneous TCP connections this listening IP has processed at any one time. Value type: UInt SNMP Name: listenIPMaxConn
total_conn	Requests sent to this listening IP. Value type: UInt SNMP Name: listenIPTotalConn

Locations

URI Path: statistics/locations/*

Locations statistics values.

Counter	Description
load	The mean load metric for this location. Value type: UInt SNMP Name: locationLoad
responses	Number of A records that have been altered to point to this location. Value type: UInt SNMP Name: locationResponses

Network interface

URI Path: `statistics/network_interface/*`

Network interface statistics values.

Counter	Description
`collisions`	The number of collisions reported by this interface. Value type: `UInt` SNMP Name: `interfaceCollisions`
`rx_bytes`	Bytes received by this interface. Value type: `UInt64` SNMP Name: `interfaceRxBytes`
`rx_bytes_hi`	Bytes received by this interface (high 32bits). Value type: `UInt` SNMP Name: `interfaceRxBytesHi`
`rx_bytes_lo`	Bytes received by this interface (low 32bits). Value type: `UInt` SNMP Name: `interfaceRxBytesLo`
`rx_errors`	The number of receive errors reported by this interface. Value type: `UInt` SNMP Name: `interfaceRxErrors`
`rx_packets`	The number of packets received by this interface.

	Value type: `UInt` SNMP Name: `interfaceRxPackets`
`tx_bytes`	Bytes transmitted by this interface. Value type: `UInt64` SNMP Name: `interfaceTxBytes`
`tx_bytes_hi`	Bytes transmitted by this interface (high 32bits). Value type: `UInt` SNMP Name: `interfaceTxBytesHi`
`tx_bytes_lo`	Bytes transmitted by this interface (low 32bits). Value type: `UInt` SNMP Name: `interfaceTxBytesLo`
`tx_errors`	The number of transmit errors reported by this interface. Value type: `UInt` SNMP Name: `interfaceTxErrors`
`tx_packets`	The number of packets transmitted by this interface. Value type: `UInt` SNMP Name: `interfaceTxPackets`

Node

URI Path: `statistics/nodes/node/*`

Node statistics values.

Counter	Description
`bytes_from_node_hi`	Bytes received from this node (high 32bits). Value type: `UInt` SNMP Name: `nodeBytesFromNodeHi`
`bytes_from_node_lo`	Bytes received from this node (low 32bits). Value type: `UInt` SNMP Name: `nodeBytesFromNodeLo`
`bytes_to_node_hi`	Bytes sent to this node (high 32bits). Value type: `UInt` SNMP Name: `nodeBytesToNodeHi`
`bytes_to_node_lo`	Bytes sent to this node (low 32bits). Value type: `UInt` SNMP Name: `nodeBytesToNodeLo`
`current_conn`	Requests currently established to this node. (does not include idle keepalives). Value type: `UInt` SNMP Name: `nodeCurrentConn`
`current_requests`	Connections currently established to this node. Value type: `UInt` SNMP `nodeCurrentRequests`

	Name:
errors	Number of timeouts, connection problems and other errors for this node. Value type: UInt SNMP Name: nodeErrors
failures	Failures of this node. Value type: UInt SNMP Name: nodeFailures
new_conn	Requests that created a new connection to this node. Value type: UInt SNMP Name: nodeNewConn
pooled_conn	Requests that reused an existing pooled/keepalive connection rather than creating a new TCP connection. Value type: UInt SNMP Name: nodePooledConn
port	The port this node listens on. Value type: UInt SNMP Name: nodePort
response_max	Maximum response time (ms) in the last second for this node. Value type: UInt SNMP Name: nodeResponseMax
response_mean	Mean response time (ms) in the last second for this node.

	Value type: `UInt` SNMP Name: `nodeResponseMean`
`response_min`	Minimum response time (ms) in the last second for this node. Value type: `UInt` SNMP Name: `nodeResponseMin`
`state`	The state of this node. Value type: `Enum(String)` SNMP Name: `nodeState` Permitted values: `alive` — alive(1) `dead` — dead(2) `unknown` — unknown(3)
`total_conn`	Requests sent to this node. Value type: `UInt` SNMP Name: `nodeTotalConn`

Node inet46

URI Path: `statistics/nodes/node_inet46/*`

Node inet46 statistics values.

Counter	Description
`bytes_from_node`	Bytes received from this node. Value type: `UInt64` SNMP Name: `nodeInet46BytesFromNode`

bytes_from_node_hi	Bytes received from this node (high 32bits). Value type: UInt SNMP Name: nodeInet46BytesFromNodeHi
bytes_from_node_lo	Bytes received from this node (low 32bits). Value type: UInt SNMP Name: nodeInet46BytesFromNodeLo
bytes_to_node	Bytes sent to this node. Value type: UInt64 SNMP Name: nodeInet46BytesToNode
bytes_to_node_hi	Bytes sent to this node (high 32bits). Value type: UInt SNMP Name: nodeInet46BytesToNodeHi
bytes_to_node_lo	Bytes sent to this node (low 32bits). Value type: UInt SNMP Name: nodeInet46BytesToNodeLo
current_conn	Current connections established to this node, includes idle connections. Value type: UInt SNMP Name: nodeInet46CurrentConn
current_requests	Active connections established to this node, does not include idle connections. Value type: UInt SNMP nodeInet46CurrentRequests

	Name:
`errors`	Number of timeouts, connection problems and other errors for this node. Value type: `UInt` SNMP Name: `nodeInet46Errors`
`failures`	Failures of this node. Value type: `UInt` SNMP Name: `nodeInet46Failures`
`idle_conns`	Number of idle HTTP connections to this node. Value type: `UInt` SNMP Name: `nodeInet46IdleConns`
`new_conn`	Requests that created a new connection to this node. Value type: `UInt` SNMP Name: `nodeInet46NewConn`
`pooled_conn`	Requests that reused an existing pooled/keepalive connection rather than creating a new TCP connection. Value type: `UInt` SNMP Name: `nodeInet46PooledConn`
`port`	The port this node listens on. Value type: `UInt` SNMP Name: `nodeInet46Port`
`response_max`	Maximum response time (ms) in the last second for this node.

	Value type: `UInt` SNMP Name: `nodeInet46ResponseMax`
`response_mean`	Mean response time (ms) in the last second for this node. Value type: `UInt` SNMP Name: `nodeInet46ResponseMean`
`response_min`	Minimum response time (ms) in the last second for this node. Value type: `UInt` SNMP Name: `nodeInet46ResponseMin`
`state`	The state of this node. Value type: `Enum(String)` SNMP Name: `nodeInet46State` Permitted values: `alive` alive(1); `dead` dead(2); `unknown` unknown(3)
`total_conn`	Requests sent to this node. Value type: `UInt` SNMP Name: `nodeInet46TotalConn`

Per location service

URI Path: `statistics/per_location_service/*`

Per location service statistics values.

Counter	Description

<table>
<tr><td>draining</td><td>The draining state of this location for this GLB Service.

Value type: Enum(String)

SNMP Name: perLocationServiceDraining

Permitted values:<table><tr><td>draining</td><td>draining(1)</td></tr><tr><td>active</td><td>active(2)</td></tr></table></td></tr>
<tr><td>frontend_state</td><td>The frontend state of this location for this GLB Service.

Value type: Enum(String)

SNMP Name: perLocationServiceFrontendState

Permitted values:<table><tr><td>alive</td><td>alive(1)</td></tr><tr><td>dead</td><td>dead(2)</td></tr></table></td></tr>
<tr><td>load</td><td>The load metric for this location for this GLB Service.

Value type: UInt

SNMP Name: perLocationServiceLoad</td></tr>
<tr><td>monitor_state</td><td>The monitor state of this location for this GLB Service.

Value type: Enum(String)

SNMP Name: perLocationServiceMonitorState

Permitted values:<table><tr><td>alive</td><td>alive(1)</td></tr><tr><td>dead</td><td>dead(2)</td></tr></table></td></tr>
<tr><td>responses</td><td>Number of A records that have been altered to point to this location for this GLB Service.

Value type: UInt

SNMP Name: perLocationServiceResponses</td></tr>
<tr><td>state</td><td>The state of this location for this GLB Service.</td></tr>
</table>

	Value type: `Enum(String)` SNMP Name: `perLocationServiceState` Permitted values:	`alive` alive(1) `dead` dead(2)

Per node service level

URI Path: `statistics/per_node_slm/per_node_service_level/*`

Per node service level statistics values.

Counter	Description
`node_port`	The port number of this node. Value type: `UInt` SNMP Name: `perNodeServiceLevelNodePort`
`response_max`	Maximum response time (ms) in the last second for this SLM class to this node. Value type: `UInt` SNMP Name: `perNodeServiceLevelResponseMax`
`response_mean`	Mean response time (ms) in the last second for this SLM class to this node. Value type: `UInt` SNMP Name: `perNodeServiceLevelResponseMean`
`response_min`	Minimum response time (ms) in the last second for this SLM class to this node. Value type: `UInt` SNMP `perNodeServiceLevelResponseMin`

	Name:
total_conn	Requests handled by this SLM class to this node. Value type: UInt SNMP Name: perNodeServiceLevelTotalConn
total_non_conf	Non-conforming requests handled by this SLM class to this node. Value type: UInt SNMP Name: perNodeServiceLevelTotalNonConf

Per node service level inet46

URI Path: statistics/per_node_slm/per_node_service_level_inet46/*

Per node service level inet46 statistics values.

Counter	Description
node_port	The port number of this node. Value type: UInt SNMP Name: perNodeServiceLevelInet46NodePort
response_max	Maximum response time (ms) in the last second for this SLM class to this node. Value type: UInt SNMP Name: perNodeServiceLevelInet46ResponseMax
response_mean	Mean response time (ms) in the last second for this SLM class to this node. Value type: UInt SNMP perNodeServiceLevelInet46ResponseMean

	Name:
response_min	Minimum response time (ms) in the last second for this SLM class to this node. Value type: UInt SNMP Name: perNodeServiceLevelInet46ResponseMin
total_conn	Requests handled by this SLM class to this node. Value type: UInt SNMP Name: perNodeServiceLevelInet46TotalConn
total_non_conf	Non-conforming requests handled by this SLM class to this node. Value type: UInt SNMP Name: perNodeServiceLevelInet46TotalNonConf

Per pool node

URI Path: statistics/nodes/per_pool_node/*

Per pool node statistics values.

Counter	Description
bytes_from_node	Bytes received from this node. Value type: UInt64 SNMP Name: perPoolNodeBytesFromNode
bytes_from_node_hi	Bytes received from this node (high 32bits). Value type: UInt SNMP Name: perPoolNodeBytesFromNodeHi

bytes_from_node_lo	Bytes received from this node (low 32bits). Value type: UInt SNMP Name: perPoolNodeBytesFromNodeLo
bytes_to_node	Bytes sent to this node. Value type: UInt64 SNMP Name: perPoolNodeBytesToNode
bytes_to_node_hi	Bytes sent to this node (high 32bits). Value type: UInt SNMP Name: perPoolNodeBytesToNodeHi
bytes_to_node_lo	Bytes sent to this node (low 32bits). Value type: UInt SNMP Name: perPoolNodeBytesToNodeLo
current_conn	Current connections established to a node, includes idle connections. Value type: UInt SNMP Name: perPoolNodeCurrentConn
current_requests	Active connections established to this node, does not include idle connections. Value type: UInt SNMP Name: perPoolNodeCurrentRequests
errors	Number of timeouts, connection problems and other errors for this node. Value type: UInt

	SNMP Name: `perPoolNodeErrors`
`failures`	Failures of this node. Value type: `UInt` SNMP Name: `perPoolNodeFailures`
`idle_conns`	Number of idle HTTP connections to this node. Value type: `UInt` SNMP Name: `perPoolNodeIdleConns`
`new_conn`	Requests that created a new connection to this node. Value type: `UInt` SNMP Name: `perPoolNodeNewConn`
`node_port`	The port that this node listens on. Value type: `UInt` SNMP Name: `perPoolNodeNodePort`
`pooled_conn`	Requests that reused an existing pooled/keepalive connection rather than creating a new TCP connection. Value type: `UInt` SNMP Name: `perPoolNodePooledConn`
`response_max`	Maximum response time (ms) in the last second for this node. Value type: `UInt` SNMP Name: `perPoolNodeResponseMax`
`response_mean`	Mean response time (ms) in the last second for this node.

	Value type: `UInt` SNMP Name: `perPoolNodeResponseMean`
`response_min`	Minimum response time (ms) in the last second for this node. Value type: `UInt` SNMP Name: `perPoolNodeResponseMin`
`state`	The state of this node. Value type: `Enum(String)` SNMP Name: `perPoolNodeState` Permitted values: `alive` — alive(1) `dead` — dead(2) `unknown` — unknown(3) `draining` — draining(4)
`total_conn`	Requests sent to this node. Value type: `UInt` SNMP Name: `perPoolNodeTotalConn`

Pools

URI Path: `statistics/pools/*`

Pools statistics values.

Counter	Description
`algorithm`	The load-balancing algorithm the pool uses. Value type: `Enum(String)`

<table>
<tr><td></td><td>SNMP Name: poolAlgorithm

Permitted values:
<table>
<tr><td>roundrobin</td><td>roundrobin(1)</td></tr>
<tr><td>weightedRoundRobin</td><td>weightedRoundRobin(2)</td></tr>
<tr><td>perceptive</td><td>perceptive(3)</td></tr>
<tr><td>leastConnections</td><td>leastConnections(4)</td></tr>
<tr><td>fastestResponseTime</td><td>fastestResponseTime(5)</td></tr>
<tr><td>random</td><td>random(6)</td></tr>
<tr><td>weightedLeastConnections</td><td>weightedLeastConnections(7)</td></tr>
</table></td></tr>
<tr><td>bytes_in</td><td>Bytes received by this pool from nodes.

Value type: UInt64

SNMP Name: poolBytesIn</td></tr>
<tr><td>bytes_in_hi</td><td>Bytes received by this pool from nodes (high 32bits).

Value type: UInt

SNMP Name: poolBytesInHi</td></tr>
<tr><td>bytes_in_lo</td><td>Bytes received by this pool from nodes (low 32bits).

Value type: UInt

SNMP Name: poolBytesInLo</td></tr>
<tr><td>bytes_out</td><td>Bytes sent by this pool to nodes.

Value type: UInt64

SNMP Name: poolBytesOut</td></tr>
<tr><td>bytes_out_hi</td><td>Bytes sent by this pool to nodes (high 32bits).

Value type: UInt</td></tr>
</table>

	SNMP Name: `poolBytesOutHi`
`bytes_out_lo`	Bytes sent by this pool to nodes (low 32bits). Value type: `UInt` SNMP Name: `poolBytesOutLo`
`conns_queued`	Total connections currently queued to this pool. Value type: `UInt` SNMP Name: `poolConnsQueued`
`disabled`	The number of nodes in this pool that are disabled. Value type: `UInt` SNMP Name: `poolDisabled`
`draining`	The number of nodes in this pool which are draining. Value type: `UInt` SNMP Name: `poolDraining`
`max_queue_time`	Maximum time a connection was queued for, over the last second. Value type: `UInt` SNMP Name: `poolMaxQueueTime`
`mean_queue_time`	Mean time a connection was queued for, over the last second. Value type: `UInt` SNMP Name: `poolMeanQueueTime`
`min_queue_time`	Minimum time a connection was queued for, over the last second. Value type: `UInt`

<table>
<tr><td></td><td>SNMP Name: <code>poolMinQueueTime</code></td></tr>
<tr><td><code>nodes</code></td><td>The number of nodes registered with this pool.
Value type: <code>UInt</code>
SNMP Name: <code>poolNodes</code></td></tr>
<tr><td><code>persistence</code></td><td>The session persistence method this pool uses
Value type: <code>Enum(String)</code>
SNMP Name: <code>poolPersistence</code>
Permitted values:
<table>
<tr><td><code>none</code></td><td>none(1)</td></tr>
<tr><td><code>ip</code></td><td>ip(2)</td></tr>
<tr><td><code>rule</code></td><td>rule(3)</td></tr>
<tr><td><code>transparent</code></td><td>transparent(4)</td></tr>
<tr><td><code>applicationCookie</code></td><td>applicationCookie(5)</td></tr>
<tr><td><code>xZeusBackend</code></td><td>xZeusBackend(6)</td></tr>
<tr><td><code>ssl</code></td><td>ssl(7)</td></tr>
</table></td></tr>
<tr><td><code>queue_timeouts</code></td><td>Total connections that timed-out while queued.
Value type: <code>UInt</code>
SNMP Name: <code>poolQueueTimeouts</code></td></tr>
<tr><td><code>session_migrated</code></td><td>Sessions migrated to a new node because the desired node was unavailable.
Value type: <code>UInt</code>
SNMP Name: <code>poolSessionMigrated</code></td></tr>
<tr><td><code>state</code></td><td>The state of this pool.
Value type: <code>Enum(String)</code>
SNMP <code>poolState</code></td></tr>
</table>

	Name: Permitted values: `active` — active(1) `disabled` — disabled(2) `draining` — draining(3) `unused` — unused(4) `unknown` — unknown(5)
`total_conn`	Requests sent to this pool. Value type: `UInt` SNMP Name: `poolTotalConn`

Rule authenticators

URI Path: `statistics/rule_authenticators/*`

Rule authenticators statistics values.

Counter	Description
`errors`	Number of connection errors that have occured when trying to connect to an authentication server. Value type: `UInt` SNMP Name: `authenticatorErrors`
`fails`	Number of times this Authenticator has failed to authenticate. Value type: `UInt` SNMP Name: `authenticatorFails`
`passes`	Number of times this Authenticator has successfully authenticated. Value type: `UInt` SNMP `authenticatorPasses`

	Name:
requests	Number of times this Authenticator has been asked to authenticate. Value type: UInt SNMP Name: authenticatorRequests

Rules

URI Path: statistics/rules/*

Rules statistics values.

Counter	Description
aborts	Number of times this TrafficScript rule has aborted. Value type: UInt SNMP Name: ruleAborts
discards	Number of times this TrafficScript rule has discarded the connection. Value type: UInt SNMP Name: ruleDiscards
executions	Number of times this TrafficScript rule has been executed. Value type: UInt SNMP Name: ruleExecutions
pool_select	Number of times this TrafficScript rule has selected a pool to use. Value type: UInt SNMP rulePoolSelect

	Name:
`responds`	Number of times this TrafficScript rule has responded directly to the client. Value type: `UInt` SNMP Name: `ruleResponds`
`retries`	Number of times this TrafficScript rule has forced the request to be retried. Value type: `UInt` SNMP Name: `ruleRetries`

Service level monitors

URI Path: `statistics/service_level_monitors/*`

Service level monitors statistics values.

Counter	Description
`conforming`	Percentage of requests associated with this SLM class that are conforming Value type: `UInt` SNMP Name: `serviceLevelConforming`
`current_conns`	The number of connections currently associated with this SLM class. Value type: `UInt` SNMP Name: `serviceLevelCurrentConns`
`is_o_k`	Indicates if this SLM class is currently conforming. Value type: `Enum(String)`

<table>
<tr><td></td><td>SNMP Name: <code>serviceLevelIsOK</code>

Permitted values:
<table>
<tr><td><code>notok</code></td><td>notok(1)</td></tr>
<tr><td><code>ok</code></td><td>ok(2)</td></tr>
</table>
</td></tr>
<tr><td><code>response_max</code></td><td>Maximum response time (ms) in the last second for this SLM class.

Value type: <code>UInt</code>

SNMP Name: <code>serviceLevelResponseMax</code></td></tr>
<tr><td><code>response_mean</code></td><td>Mean response time (ms) in the last second for this SLM class.

Value type: <code>UInt</code>

SNMP Name: <code>serviceLevelResponseMean</code></td></tr>
<tr><td><code>response_min</code></td><td>Minimum response time (ms) in the last second for this SLM class.

Value type: <code>UInt</code>

SNMP Name: <code>serviceLevelResponseMin</code></td></tr>
<tr><td><code>total_conn</code></td><td>Requests handled by this SLM class.

Value type: <code>UInt</code>

SNMP Name: <code>serviceLevelTotalConn</code></td></tr>
<tr><td><code>total_non_conf</code></td><td>Non-conforming requests handled by this SLM class.

Value type: <code>UInt</code>

SNMP Name: <code>serviceLevelTotalNonConf</code></td></tr>
</table>

Service protection

URI Path: `statistics/service_protection/*`

Service protection statistics values.

Counter	Description
`last_refusal_time`	The time (in hundredths of a second) since this service protection class last refused a connection (this value will wrap if no connections are refused in more than 497 days). Value type: `UInt` SNMP Name: `serviceProtLastRefusalTime`
`refusal_binary`	Connections refused by this service protection class because the request contained disallowed binary content. Value type: `UInt` SNMP Name: `serviceProtRefusalBinary`
`refusal_conc10_ip`	Connections refused by this service protection class because the top 10 source IP addresses issued too many concurrent connections. Value type: `UInt` SNMP Name: `serviceProtRefusalConc10IP`
`refusal_conc1_ip`	Connections refused by this service protection class because the source IP address issued too many concurrent connections. Value type: `UInt` SNMP Name: `serviceProtRefusalConc1IP`
`refusal_conn_rate`	Connections refused by this service protection class because the source IP address issued too many connections within 60 seconds. Value type: `UInt` SNMP Name: `serviceProtRefusalConnRate`

refusal_ip	Connections refused by this service protection class because the source IP address was banned. Value type: UInt SNMP Name: serviceProtRefusalIP
refusal_rfc2396	Connections refused by this service protection class because the HTTP request was not RFC 2396 compliant. Value type: UInt SNMP Name: serviceProtRefusalRFC2396
refusal_size	Connections refused by this service protection class because the request was larger than the defined limits allowed. Value type: UInt SNMP Name: serviceProtRefusalSize
total_refusal	Connections refused by this service protection class. Value type: UInt SNMP Name: serviceProtTotalRefusal

Ssl cache

URI Path: statistics/cache/ssl_cache

Ssl cache statistics values.

Counter	Description
entries	The total number of SSL sessions stored in the server cache. Value type: UInt SNMP Name: sslCacheEntries

entries_max	The maximum number of SSL entries in the server cache. Value type: UInt SNMP Name: sslCacheEntriesMax
hit_rate	The percentage of SSL server cache lookups that succeeded. Value type: UInt SNMP Name: sslCacheHitRate
hits	Number of times a SSL entry has been successfully found in the server cache. Value type: UInt SNMP Name: sslCacheHits
lookups	Number of times a SSL entry has been looked up in the server cache. Value type: UInt SNMP Name: sslCacheLookups
misses	Number of times a SSL entry has not been available in the server cache. Value type: UInt SNMP Name: sslCacheMisses
oldest	The age of the oldest SSL session in the server cache (in seconds). Value type: UInt SNMP Name: sslCacheOldest

Ssl ocsp stapling

URI Path: `statistics/ssl_ocsp_stapling`

Ssl ocsp stapling statistics values.

Counter	Description
`cache_count`	The number of entries in the OCSP stapling cache. Value type: `UInt` SNMP Name: `sslOcspStaplingCacheCount`
`count`	The number of outgoing OCSP requests for OCSP stapling. Value type: `UInt` SNMP Name: `sslOcspStaplingCount`
`failure_count`	The number of failed outgoing OCSP requests for OCSP stapling. Value type: `UInt` SNMP Name: `sslOcspStaplingFailureCount`
`good_count`	The number of 'good' OCSP responses for OCSP stapling. Value type: `UInt` SNMP Name: `sslOcspStaplingGoodCount`
`revoked_count`	The number of 'revoked' OCSP responses for OCSP stapling. Value type: `UInt` SNMP Name: `sslOcspStaplingRevokedCount`
`success_count`	The number of successful outgoing OCSP requests for OCSP stapling. Value type: `UInt` SNMP `sslOcspStaplingSuccessCount`

	Name:
unknown_count	The number of 'unknown' OCSP requests for OCSP stapling. Value type: UInt SNMP Name: sslOcspStaplingUnknownCount

Ssl session cache

URI Path: statistics/cache/ssl_session_cache

Ssl session cache statistics values.

Counter	Description
entries	The total number of SSL session persistence entries stored in the cache. Value type: UInt SNMP Name: sslSessionCacheEntries
entries_max	The maximum number of SSL session persistence entries in the cache. Value type: UInt SNMP Name: sslSessionCacheEntriesMax
hit_rate	The percentage of SSL session persistence lookups that succeeded. Value type: UInt SNMP Name: sslSessionCacheHitRate
hits	Number of times a SSL session persistence entry has been successfully found in the cache. Value type: UInt

	SNMP Name: `sslSessionCacheHits`
`lookups`	Number of times a SSL session persistence entry has been looked up in the cache. Value type: `UInt` SNMP Name: `sslSessionCacheLookups`
`misses`	Number of times a SSL session persistence entry has not been available in the cache. Value type: `UInt` SNMP Name: `sslSessionCacheMisses`
`oldest`	The age of the oldest SSL session in the cache (in seconds). Value type: `UInt` SNMP Name: `sslSessionCacheOldest`

Traffic ip

URI Path: `statistics/traffic_ips/traffic_ip/*`

Traffic ip statistics values.

Counter	Description
`state`	Whether this traffic IP address is currently being hosted by this traffic manager. Value type: `Enum(String)` SNMP Name: `trafficIPState` Permitted values: `raised` raised(1); `lowered` lowered(2)

time	The time (in hundredths of a second) since trafficIPState last changed (this value will wrap if the state hasn't changed for 497 days). Value type: UInt SNMP Name: trafficIPTime

Traffic ip inet46

URI Path: statistics/traffic_ips/traffic_ip_inet46/*

Traffic ip inet46 statistics values.

Counter	Description
state	Whether this traffic IP address is currently being hosted by this traffic manager. Value type: Enum(String) SNMP Name: trafficIPInet46State Permitted values: raised(1), lowered(2)
time	The time (in hundredths of a second) since trafficIPState last changed (this value will wrap if the state hasn't changed for 497 days). Value type: UInt SNMP Name: trafficIPInet46Time

Uni session cache

URI Path: `statistics/cache/uni_session_cache`

Uni session cache statistics values.

Counter	Description
`entries`	The total number of universal sessions stored in the cache. Value type: `UInt` SNMP Name: `uniSessionCacheEntries`
`entries_max`	The maximum number of universal sessions in the cache. Value type: `UInt` SNMP Name: `uniSessionCacheEntriesMax`
`hit_rate`	The percentage of universal session lookups that succeeded. Value type: `UInt` SNMP Name: `uniSessionCacheHitRate`
`hits`	Number of times a universal session entry has been successfully found in the cache. Value type: `UInt` SNMP Name: `uniSessionCacheHits`
`lookups`	Number of times a universal session entry has been looked up in the cache. Value type: `UInt` SNMP Name: `uniSessionCacheLookups`
`misses`	Number of times a universal session entry has not been available in the cache. Value type: `UInt`

	SNMP Name: `uniSessionCacheMisses`
`oldest`	The age of the oldest universal session in the cache (in seconds). Value type: `UInt` SNMP Name: `uniSessionCacheOldest`

Virtual servers

URI Path: `statistics/virtual_servers/*`

Virtual servers statistics values.

Counter	Description
`bytes_in`	Bytes received by this virtual server from clients. Value type: `UInt64` SNMP Name: `virtualserverBytesIn`
`bytes_in_hi`	Bytes received by this virtual server from clients (high 32bits). Value type: `UInt` SNMP Name: `virtualserverBytesInHi`
`bytes_in_lo`	Bytes received by this virtual server from clients (low 32bits). Value type: `UInt` SNMP Name: `virtualserverBytesInLo`
`bytes_out`	Bytes sent by this virtual server to clients. Value type: `UInt64` SNMP Name: `virtualserverBytesOut`

bytes_out_hi	Bytes sent by this virtual server to clients (high 32bits). Value type: UInt SNMP Name: virtualserverBytesOutHi
bytes_out_lo	Bytes sent by this virtual server to clients (low 32bits). Value type: UInt SNMP Name: virtualserverBytesOutLo
cert_status_requests	Number of incoming TLS handshakes for this virtual server with certificate status requests. Value type: UInt SNMP Name: virtualserverCertStatusRequests
cert_status_responses	Number of incoming TLS handshakes for this virtual server to which certificate status responses were attached. Value type: UInt SNMP Name: virtualserverCertStatusResponses
connect_timed_out	Connections closed by this virtual server because the 'connect_timeout' interval was exceeded. Value type: UInt SNMP Name: virtualserverConnectTimedOut
connection_errors	Number of transaction or protocol errors in this virtual server. Value type: UInt SNMP Name: virtualserverConnectionErrors
connection_failures	Number of connection failures in this virtual server. Value type: UInt

	SNMP Name: `virtualserverConnectionFailures`
`current_conn`	TCP connections currently established to this virtual server. Value type: `UInt` SNMP Name: `virtualserverCurrentConn`
`data_timed_out`	Connections closed by this virtual server because the 'timeout' interval was exceeded. Value type: `UInt` SNMP Name: `virtualserverDataTimedOut`
`direct_replies`	Direct replies from this virtual server, without forwarding to a node. Value type: `UInt` SNMP Name: `virtualserverDirectReplies`
`discard`	Connections discarded by this virtual server. Value type: `UInt` SNMP Name: `virtualserverDiscard`
`gzip`	Responses which have been compressed by content compression. Value type: `UInt` SNMP Name: `virtualserverGzip`
`gzip_bytes_saved`	Bytes of network traffic saved by content compression. Value type: `UInt64` SNMP Name: `virtualserverGzipBytesSaved`
`gzip_bytes_saved_hi`	Bytes of network traffic saved by content compression (high 32bits

	). Value type: `UInt` SNMP Name: `virtualserverGzipBytesSavedHi`
`gzip_bytes_saved_lo`	Bytes of network traffic saved by content compression (low 32bits). Value type: `UInt` SNMP Name: `virtualserverGzipBytesSavedLo`
`http_cache_hit_rate`	Percentage hit rate of the web cache for this virtual server. Value type: `UInt` SNMP Name: `virtualserverHttpCacheHitRate`
`http_cache_hits`	HTTP responses sent directly from the web cache by this virtual server. Value type: `UInt` SNMP Name: `virtualserverHttpCacheHits`
`http_cache_lookups`	HTTP requests that are looked up in the web cache by this virtual server. Value type: `UInt` SNMP Name: `virtualserverHttpCacheLookups`
`http_rewrite_cookie`	HTTP Set-Cookie headers, supplied by a node, that have been rewritten. Value type: `UInt` SNMP Name: `virtualserverHttpRewriteCookie`
`http_rewrite_location`	HTTP Location headers, supplied by a node, that have been rewritten.

	Value type: `UInt` SNMP Name: `virtualserverHttpRewriteLocation`
`keepalive_timed_out`	Connections closed by this virtual server because the 'keepalive_timeout' interval was exceeded. Value type: `UInt` SNMP Name: `virtualserverKeepaliveTimedOut`
`max_conn`	Maximum number of simultaneous TCP connections this virtual server has processed at any one time. Value type: `UInt` SNMP Name: `virtualserverMaxConn`
`port`	The port the virtual server listens on. Value type: `UInt` SNMP Name: `virtualserverPort`
`protocol`	The protocol the virtual server is operating. Value type: `Enum(String)` SNMP Name: `virtualserverProtocol` Permitted values:

`http`	http(1)
`https`	https(2)
`ftp`	ftp(3)
`imaps`	imaps(4)
`imapv2`	imapv2(5)
`imapv3`	imapv3(6)
`imapv4`	imapv4(7)
`pop3`	pop3(8)

pop3s	pop3s(9)
smtp	smtp(10)
ldap	ldap(11)
ldaps	ldaps(12)
telnet	telnet(13)
sslforwarding	sslforwarding(14)
udpstreaming	udpstreaming(15)
udp	udp(16)
dns	dns(17)
genericserverfirst	genericserverfirst(18)
genericclientfirst	genericclientfirst(19)
dnstcp	dnstcp(20)
sipudp	sipudp(21)
siptcp	siptcp(22)
rtsp	rtsp(23)

sip_rejected_requests	Number of SIP requests rejected due to them exceeding the maximum amount of memory allocated to the connection. Value type: UInt SNMP Name: virtualserverSIPRejectedRequests
sip_total_calls	Total number of SIP INVITE requests seen by this virtual server. Value type: UInt SNMP Name: virtualserverSIPTotalCalls
total_conn	Requests received by this virtual server. Value type: UInt SNMP Name: virtualserverTotalConn

total_dgram	UDP datagrams processed by this virtual server. Value type: UInt SNMP Name: virtualserverTotalDgram
udp_timed_out	Connections closed by this virtual server because the 'udp_timeout' interval was exceeded. Value type: UInt SNMP Name: virtualserverUdpTimedOut

Web cache

URI Path: statistics/cache/web_cache

Web cache statistics values.

Counter	Description
entries	The number of items in the web cache. Value type: UInt SNMP Name: webCacheEntries
hit_rate	The percentage of web cache lookups that succeeded. Value type: UInt SNMP Name: webCacheHitRate
hits	Number of times a page has been successfully found in the web cache. Value type: UInt64 SNMP Name: webCacheHits

hits_hi	Number of times a page has been successfully found in the web cache (high 32 bits). Value type: UInt SNMP Name: webCacheHitsHi
hits_lo	Number of times a page has been successfully found in the web cache (low 32 bits). Value type: UInt SNMP Name: webCacheHitsLo
lookups	Number of times a page has been looked up in the web cache. Value type: UInt64 SNMP Name: webCacheLookups
lookups_hi	Number of times a page has been looked up in the web cache (high 32 bits). Value type: UInt SNMP Name: webCacheLookupsHi
lookups_lo	Number of times a page has been looked up in the web cache (low 32 bits). Value type: UInt SNMP Name: webCacheLookupsLo
max_entries	The maximum number of items in the web cache. Value type: UInt SNMP Name: webCacheMaxEntries
mem_maximum	The maximum amount of memory the web cache can use in kilobytes.

	Value type: `UInt` SNMP Name: `webCacheMemMaximum`
`mem_used`	Total memory used by the web cache in kilobytes. Value type: `UInt` SNMP Name: `webCacheMemUsed`
`misses`	Number of times a page has not been found in the web cache. Value type: `UInt64` SNMP Name: `webCacheMisses`
`misses_hi`	Number of times a page has not been found in the web cache (high 32 bits). Value type: `UInt` SNMP Name: `webCacheMissesHi`
`misses_lo`	Number of times a page has not been found in the web cache (low 32 bits). Value type: `UInt` SNMP Name: `webCacheMissesLo`
`oldest`	The age of the oldest item in the web cache (in seconds). Value type: `UInt` SNMP Name: `webCacheOldest`

System Information Resources

Information

URI Path: `information`

Static information for the system.

Property	Description
`tm_version`	Version number of the Traffic Manager instance. Value type: `String`

CHAPTER 5 Further Information and Resources

Stingray Manuals

Your traffic management system includes an **Installation and Getting Started Guide**, intended to get you up and running quickly, and a more detailed **User Manual**. There are also full reference manuals for functionality such as the Java Extensions and TrafficScript.

You can access these manuals via the **Help** pages (described below), or download the most recent versions from the Riverbed Support website at:

https://support.riverbed.com/software/index.htm

Information Online

Product specifications can be found at:

http://www.riverbed.com/products-solutions/products/application-delivery-stingray/

Visit the Riverbed Splash community website for further documentation, examples, white papers, and other resources:

http://splash.riverbed.com

APPENDIX A Changes in this Version of the API

Stingray Traffic Manager 9.5 includes version 2.0 of the REST API. This appendix lists the differences between this version of the API and the previous released version (1.1).

Applications that were developed against older versions of the Traffic Manager REST API might be affected. Use the information contained in this chapter to identify the necessary updates.

The following resources were added:

- Resource 'Actions' (/api/tm/2.0/status/local_tm/statistics/actions/*)
- Resource 'Asp session cache' (/api/tm/2.0/status/local_tm/statistics/cache/asp_session_cache)
- Resource 'Bandwidth' (/api/tm/2.0/status/local_tm/statistics/bandwidth/*)
- Resource 'Cloud api credentials' (/api/tm/2.0/status/local_tm/statistics/cloud_api_credentials/*)
- Resource 'Connection rate limit' (/api/tm/2.0/status/local_tm/statistics/connection_rate_limit/*)
- Resource 'Custom configuration set' (/api/tm/2.0/config/active/custom)
- Resource 'Events' (/api/tm/2.0/status/local_tm/statistics/events/*)
- Resource 'Glb services' (/api/tm/2.0/status/local_tm/statistics/glb_services/*)
- Resource 'Globals' (/api/tm/2.0/status/local_tm/statistics/globals)
- Resource 'Information' (/api/tm/2.0/status/local_tm/information)
- Resource 'Ip gateway' (/api/tm/2.0/status/local_tm/statistics/traffic_ips/ip_gateway)
- Resource 'Ip session cache' (/api/tm/2.0/status/local_tm/statistics/cache/ip_session_cache)
- Resource 'J2ee session cache' (/api/tm/2.0/status/local_tm/statistics/cache/j2ee_session_cache)
- Resource 'Listen ips' (/api/tm/2.0/status/local_tm/statistics/listen_ips/*)
- Resource 'Locations' (/api/tm/2.0/status/local_tm/statistics/locations/*)
- Resource 'Network interface' (/api/tm/2.0/status/local_tm/statistics/network_interface/*)
- Resource 'Node inet46' (/api/tm/2.0/status/local_tm/statistics/nodes/node_inet46/*)
- Resource 'Node' (/api/tm/2.0/status/local_tm/statistics/nodes/node/*)
- Resource 'Per location service' (/api/tm/2.0/status/local_tm/statistics/per_location_service/*)
- Resource 'Per node service level inet46' (/api/tm/2.0/status/local_tm/statistics/per_node_slm/per_node_service_level_inet46/*)
- Resource 'Per node service level' (/api/tm/2.0/status/local_tm/statistics/per_node_slm/per_node_service_level/*)
- Resource 'Per pool node' (/api/tm/2.0/status/local_tm/statistics/nodes/per_pool_node/*)

- Resource 'Pools' (/api/tm/2.0/status/local_tm/statistics/pools/*)
- Resource 'Rule authenticators' (/api/tm/2.0/status/local_tm/statistics/rule_authenticators/*)
- Resource 'Rules' (/api/tm/2.0/status/local_tm/statistics/rules/*)
- Resource 'Service level monitors' (/api/tm/2.0/status/local_tm/statistics/service_level_monitors/*)
- Resource 'Service protection' (/api/tm/2.0/status/local_tm/statistics/service_protection/*)
- Resource 'Ssl cache' (/api/tm/2.0/status/local_tm/statistics/cache/ssl_cache)
- Resource 'Ssl ocsp stapling' (/api/tm/2.0/status/local_tm/statistics/ssl_ocsp_stapling)
- Resource 'Ssl session cache' (/api/tm/2.0/status/local_tm/statistics/cache/ssl_session_cache)
- Resource 'Traffic ip inet46' (/api/tm/2.0/status/local_tm/statistics/traffic_ips/traffic_ip_inet46/*)
- Resource 'Traffic ip' (/api/tm/2.0/status/local_tm/statistics/traffic_ips/traffic_ip/*)
- Resource 'Uni session cache' (/api/tm/2.0/status/local_tm/statistics/cache/uni_session_cache)
- Resource 'Virtual servers' (/api/tm/2.0/status/local_tm/statistics/virtual_servers/*)
- Resource 'Web cache' (/api/tm/2.0/status/local_tm/statistics/cache/web_cache)

The paths of the following resources have been changed:

- Resource 'Action Program' changed path from '/api/tm/1.1/config/active/actionprogs' to '/api/tm/2.0/config/active/action_programs'.
- Resource 'Cloud Credentials' changed path from '/api/tm/1.1/config/active/cloudcredentials' to '/api/tm/2.0/config/active/cloud_api_credentials'.
- Resource 'Event Type' changed path from '/api/tm/1.1/config/active/events' to '/api/tm/2.0/config/active/event_types'.
- Resource 'Extra File' changed path from '/api/tm/1.1/config/active/extra' to '/api/tm/2.0/config/active/extra_files'.
- Resource 'GLB Service' changed path from '/api/tm/1.1/config/active/services' to '/api/tm/2.0/config/active/glb_services'.
- Resource 'Global Settings' changed path from '/api/tm/1.1/config/active/settings.cfg' to '/api/tm/2.0/config/active/global_settings'.
- Resource 'License' changed path from '/api/tm/1.1/config/active/licensekeys' to '/api/tm/2.0/config/active/license_keys'.
- Resource 'Monitor Program' changed path from '/api/tm/1.1/config/active/scripts' to '/api/tm/2.0/config/active/monitor_scripts'.
- Resource 'SLM Class' changed path from '/api/tm/1.1/config/active/slm' to '/api/tm/2.0/config/active/service_level_monitors'.

Changes in this Version of the API

- Resource 'Traffic IP Group' changed path from '/api/tm/1.1/config/active/flipper' to '/api/tm/2.0/config/active/traffic_ip_groups'.
- Resource 'Traffic Manager' changed path from '/api/tm/1.1/config/active/zxtms' to '/api/tm/2.0/config/active/traffic_managers'.
- Resource 'TrafficScript Authenticator' changed path from '/api/tm/1.1/config/active/authenticators' to '/api/tm/2.0/config/active/rule_authenticators'.
- Resource 'User Authenticator' changed path from '/api/tm/1.1/config/active/auth' to '/api/tm/2.0/config/active/user_authenticators'.
- Resource 'User Group' changed path from '/api/tm/1.1/config/active/groups' to '/api/tm/2.0/config/active/user_groups'.
- Resource 'Virtual Server' changed path from '/api/tm/1.1/config/active/vservers' to '/api/tm/2.0/config/active/virtual_servers'.

The following properties in 'Pool' (/api/tm/2.0/config/active/pools) have been added:

- Property 'basic/max_connection_attempts' was added.
- Property 'basic/max_timed_out_connection_attempts' was added.

The following properties in 'Global Settings' (/api/tm/2.0/config/active/global_settings) have been added:

- Property 'admin/ssl3_min_rehandshake_interval' was added.
- Property 'appliance/return_path_routing_enabled' was added.
- Property 'aptimizer/default_profile' was added.
- Property 'aptimizer/default_scope' was added.
- Property 'ec2/vpc_decluster_on_stop' was added.
- Property 'ip/appliance_returnpath' was added.
- Property 'ssl/ocsp_stapling_default_refresh_interval' was added.
- Property 'ssl/ocsp_stapling_mem_size' was added.
- Property 'ssl/ocsp_stapling_minimum_refresh_interval' was added.
- Property 'ssl/ocsp_stapling_prefetch' was added.
- Property 'ssl/ssl3_min_rehandshake_interval' was added.

The following properties in 'Alerting Action' (/api/tm/2.0/config/active/actions) have been added:

- Property 'basic/syslog_msg_len_limit' was added.

The following properties in 'Traffic Manager' (/api/tm/2.0/config/active/traffic_managers) have been added:

- Property 'appliance/manageec2conf' was added.

- Property 'appliance/manageiptrans' was added.
- Property 'appliance/managereturnpath' was added.
- Property 'appliance/managesysctl' was added.
- Property 'appliance/managevpcconf' was added.
- Property 'basic/appliance_sysctl' was added.
- Property 'ec2/availability_zone' was added.
- Property 'ec2/instanceid' was added.
- Property 'ec2/vpcid' was added.

The following properties in 'Virtual Server' (/api/tm/2.0/config/active/virtual_servers) have been added:

- Property 'ssl/ocsp_stapling' was added.
- Property 'syslog/msg_len_limit' was added.

The following properties in 'Traffic Manager' (/api/tm/2.0/config/active/traffic_managers) have been removed:

- Property 'appliance/ncss_nethsm' was removed.
- Property 'appliance/ncss_nethsm_esn' was removed.
- Property 'appliance/ncss_nethsm_hash' was removed.
- Property 'appliance/ncss_rfs' was removed.

The type of the following properties in 'Global Settings' (/api/tm/2.0/config/active/global_settings) has changed:

- Property 'cluster_comms/allowed_update_hosts' changed type from 'String' to 'List of strings'.

The type of the following properties in 'Traffic Manager' (/api/tm/2.0/config/active/traffic_managers) has changed:

- Property 'appliance/name_servers' changed type from 'String' to 'Set of strings (array of unique strings)'.
- Property 'appliance/ntpservers' changed type from 'String' to 'List of strings'.
- Property 'appliance/search_domains' changed type from 'String' to 'Set of strings (array of unique strings)'.