Stingray Traffic Manager
User Manual

Version 9.0

June 2012

Copyright Notice

Contents

CHAPTER 1 Introduction

Introducing This Manual

This manual contains instructions for advanced configuration of the Riverbed Stingray traffic management family of solutions. You will also find the **Installation and Getting Started Guide** useful as an introduction to installing Stingray and performing basic configuration to load-balance services.

There are several versions of the **Installation and Getting Started Guide**, and you should refer to the version appropriate for the Stingray product variant you are installing.

This manual describes the functionality of, and uses screenshots from, version 9.0.

Introducing the Stingray traffic management product family

Stingray provides high-availability, application-centric traffic management and load balancing solutions. These solutions provide control, intelligence, security and resilience for all your application traffic. Stingray traffic management is 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.

Stingray's unique process architecture ensures it can handle large volumes of network traffic efficiently. Its TrafficCluster™ scalability allows you to add more front-end traffic managers or back-end servers to your cluster as the need arises. The cluster size is unlimited, and the performance of the traffic manager grows in line with the performance of the platform used.

This manual deals primarily with single-site traffic manager clusters. Multi-site management can be enabled in your infrastructure should you wish to coordinate and manage multiple geographically-distributed clusters. This is covered fully in CHAPTER 28.

Stingray traffic management products are highly capable solutions that can also be adapted and extended as new requirements arise. Using the TrafficScript language and Java Extensions you can write sophisticated, tailored traffic management rules to inspect, transform, manage and route requests and responses. TrafficScript rules can manage connections in any TCP or UDP-based protocol.

Stingray traffic management products are secure out-of-the-box, and are hardened against intrusion and Denial-of-Service (DoS) attacks. They incorporate the fastest and strongest SSL encryption technologies, and can efficiently decrypt and encrypt large numbers of SSL connections. TrafficScript rules, security policies and other content-based calculations can be applied to the encrypted request while retaining full end-to-end security.

For critical, high-availability solutions, Stingray offers TrafficCluster™ redundancy. This allows you to have unlimited numbers of active and passive standby front-end servers. If one of your active machines fails, a standby server is automatically brought into action; in the case of subsequent failure, more standby servers are available to take up the load. This ensures that there is no single point of failure in the system.

Typical Deployment

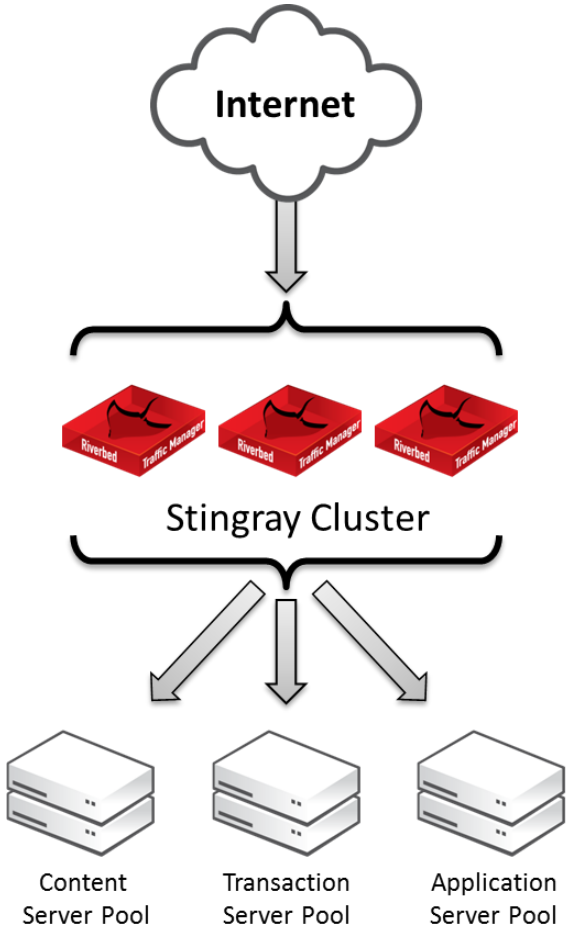


Fig. 1. A typical deployment of Stingray traffic management using a cluster

Stingray traffic manager product versions

Stingray traffic management is available in a variety of software, virtual appliance and cloud instance configurations. All variants share the same core Stingray software, but different versions may provide different levels of functionality depending on the enabling license key. Where no license key is provided, the traffic manager operates in Developer mode (see below).

This manual documents the full functionality of the Stingray traffic management software with all options enabled. It may describe features and capabilities that are not present or visible in the version of the product you are using. Features present but not enabled in your license key will be greyed-out and un-selectable in the Admin UI.

For example, Global Load Balancing, Service Level Monitoring, Rate Shaping, Auto-Scaling and Bandwidth Management are examples of advanced product capabilities and may not be enabled in your particular configuration.

Additionally, Stingray provides two major optional components that are only available through an appropriate license key upgrade:

- **Stingray Application Firewall** is an implementation of a web application firewall, used for providing advanced attack detection and protection for your web applications. Please refer to *The Stingray Application Firewall* section of CHAPTER 7 for more details of how this fits into your traffic manager infrastructure. You can also find full instructions and details in the *Stingray Application Firewall User Guide*, available from the Riverbed Support website at:

  https://support.riverbed.com/docs/stingray/index.htm

- **Stingray Aptimizer** provides content optimization functionality for your web applications. This is available as either a fully integrated component of the traffic manager, or in standalone proxy mode whereby the load balancing aspects of Stingray are disabled. Please discuss with your sales representative which variant is most appropriate for your needs. CHAPTER 20 provides full details of how to enable and configure Aptimizer for your infrastructure.

  Note: Stingray Aptimizer functionality is not currently available for Solaris/SunOS software variants.

Appliance/cloud versions of the product feature Networking and Date/Time configuration options not available in software-only versions.

Your product version specifications will describe which capabilities are enabled in your particular variant. The supplied **Installation and Getting Started Guide** will also provide further details.

Developer mode

When unlicensed, Stingray falls back to a default state known as Developer mode. This is designed to allow the user to experience the full features and capabilities of the traffic manager for development or evaluation purposes. Full product functionality is provided, but in a bandwidth-constrained environment. It operates with a maximum bandwidth limited to 1Mb/s and 100 SSL TPS (transactions per second).

Important: The Developer mode is not designed or intended for full production use. It is recommended that you contact your support provider for details of how to purchase a license key suitable for your needs.

Supported platforms

The Stingray traffic management software can be deployed on a range of platforms, on physical or virtual servers, and in cloud infrastructures. Please refer to the release notes and documentation at https://support.riverbed.com/docs/stingray/index.htm for up-to-date platform and version number requirements.

Stingray traffic management software

Stingray is available as a software-only package suitable for deployment on existing linux/unix servers. Supported distributions are listed in the release notes as mentioned above.

Virtual Appliances

Riverbed provides Stingray Virtual Appliance packages for VMware vSphere, Citrix XenServer, OracleVM and Steelhead RSP. These appliance images contain the Stingray traffic management software package pre-installed with a bespoke linux operating system layer.

Cloud Computing

A range of cloud environments are supported by Stingray, such as Amazon's Elastic Compute Cloud (EC2). The software is available on EC2 as an Amazon Machine Image (AMI), and instances are charged by the hour using the DevPay payment system. Standard and premium support is also available using DevPay. Please visit http://www.zeus.com/downloads/developers/ec2 for more information on Stingray and EC2.

Supported cluster combinations

As discussed in previous sections, Stingray is available as a number of product variants on a selection of platforms. In addition to the core Stingray software, various product options and capabilities are available through a suitable license upgrade.

When clustering multiple traffic managers together for high-availability and redundancy, it is possible to take advantage of Stingray's flexible architecture and host instances on different platforms within the same cluster. However, some care must be taken to ensure that each cluster member is running the same product configuration.

The following should be taken into account when planning your cluster:

- Mixing 32-bit and 64-bit versions of the software variant is supported and should not adversely affect cluster operations. The same is also true of multiple alternate host operating systems;

- Whilst mixing license key types IS technically possible within the cluster, it is not recommended. There are likely to be warnings and/or errors within the Admin UI if features only licensed on a sub-set of the cluster are used. This is due to the automatic config replicator attempting to apply non-licensed functionality across the cluster;

- Cloud instances (e.g. Stingray on Amazon EC2) cannot be clustered with non-cloud instances (software, virtual appliance, hardware appliance);

- The Stingray Application Firewall feature cannot be used in a mixed-platform cluster (e.g. software and virtual appliance);

- Having different product versions within a cluster is unsupported, except in the case of performing an upgrade (or downgrade) of the entire cluster. During this transient state, some traffic managers will report that they are running different versions to others in the cluster. This is to be expected, and you should find that such error conditions are cleared once all cluster members have been upgraded.

Important: If you are in any doubt as to the potential effects of running your proposed Stingray infrastructure, it is strongly recommended you contact your support provider for assistance.

Chapter Outline

CHAPTER 2 Network Layouts

This chapter discusses the network configuration you will need in a successful traffic-managed server farm. It covers hardware requirements, the layout of the network, IP addresses and DNS entries.

CHAPTER 3 Initial Configuration

This chapter explains the concepts of the Stingray traffic management architecture, including its system of *Virtual Servers*, *Pools* and *Rules*. It describes how to start managing your first service.

CHAPTER 4 Virtual Servers

Virtual Servers are one of the two fundamental configuration concepts in the Stingray software. This chapter describes Virtual Servers in more detail.

CHAPTER 5 Pools

A Pool is the second of the two fundamental configuration concepts in the Stingray software. This chapter describes Pools in more detail.

CHAPTER 6 Traffic IP Groups and Fault Tolerance

Fault Tolerance – resilience to failures of back-end servers or individual traffic manager systems – is covered in this chapter.

CHAPTER 7 Key features in the Stingray Administration Interface

Now that you have a basic traffic manager system running, this chapter covers Stingray's performance monitoring and diagnosis functions and other tools in the Stingray Administration Interface.

CHAPTER 8 TrafficScript Rules

This chapter introduces the TrafficScript language, used for writing rules to manage your traffic. It explains how to create rules and gives some examples. A separate **TrafficScript Manual** describes Stingray's TrafficScript capability in full detail.

CHAPTER 9 TrafficScript Authentication Support

This chapter provides details of remote authentication support through TrafficScript. This can be used to add remote user verification to your virtual servers.

CHAPTER 10 Java Extensions

This chapter describes Stingray's Java Extensions, allowing you to invoke Java code from TrafficScript. A separate Java Development Guide describes Stingray's Java Extensions capability in full detail.

CHAPTER 11 Protocol support

Some protocols can benefit from specific support, and this chapter describes special protocol handlers in Stingray traffic management products.

CHAPTER 12 Session Persistence

This chapter describes Stingray's session persistence features. It outlines some useful session persistence strategies and explains how it is set up within the product.

CHAPTER 13 SSL Encryption

This chapter deals with SSL decryption of incoming traffic, and encryption of requests being sent to the back-end servers. It describes how to set up SSL certificates, authorities and revocation lists within the product and how to apply them to your services.

CHAPTER 14 Health Monitoring

This chapter discusses Stingray's Health Monitoring capabilities. Health Monitors are used to test the correct operation of back-end server nodes; in the event of a failure, they cause the traffic manager to raise an alert, and to route traffic away from the failed node.

CHAPTER 15 Service Protection

This chapter describes the service protection capabilities of the product. It explains how a class of service protection settings can be defined in the catalog, and used by one or more services.

CHAPTER 16 Bandwidth Management

This chapter describes the bandwidth management capabilities of the product. It explains how a class of bandwidth management settings can be defined in the catalog, and used by one or more services.

CHAPTER 17 Request Rate Shaping

This chapter describes how a traffic manager can be used to rate-shape requests to your servers and applications to restrict users' activities and prevent applications from being overwhelmed by requests.

CHAPTER 18 Service Level Monitoring

This chapter describes the service level monitoring of the product. It explains how a class of service level monitoring settings can be defined in the catalog, and used by one or more services.

CHAPTER 19 Content Caching

This chapter describes Stingray's HTTP Content Caching capabilities. You can cache commonly-requested web resources and respond to subsequent requests directly

from the cache, rather than forwarding many identical requests to the back-end server nodes.

CHAPTER 20 Aptimizing your web content

This chapter covers the Stingray Aptimizer and how you can apply web content optimization to your web services.

CHAPTER 21 Event Handling and Alerts

This chapter describes how to control what the traffic manager does when particular events occur, and how to inform or drive external management systems.

CHAPTER 22 Configuring Stingray in a virtual or cloud environment

This chapter describes the additional configuration tasks specific to Stingray as a virtual appliance or cloud instance. It is not applicable to Stingray traffic management software variants.

CHAPTER 23 System Security

This chapter gives tips for secure operation of your traffic managers. It covers firewalling and network design, and Unix software permissions which are relevant to the traffic manager software.

CHAPTER 24 Admin Server Security

This chapter covers security measures to control access to the Administration Interface, including user authentication against external databases.

CHAPTER 25 Stingray Control API

This chapter gives a very brief introduction to the Stingray Control API. For full information, please refer to the accompanying **Stingray Control API Manual**.

CHAPTER 26 Command Line Interface

This chapter covers the Command Line Interface in detail, with reference to the Control API methods defined in the accompanying **Stingray Control API Manual**.

CHAPTER 27 Granular Configuration Import/Export with zconf

This chapter describes the `zconf` command-line utility which can be used to perform a fine-grained import/export on individual configuration objects.

CHAPTER 28 Multi-site cluster management

Furthering the concept of a cluster of traffic managers, this chapter introduces the idea of combining several clusters into one centrally-managed super-, or *multi-site*, cluster.

CHAPTER 29 Global Load Balancing

This chapter discusses the implementation of Global Server Load Balancing (GSLB) within Stingray.

CHAPTER 30 Troubleshooting

This chapter describes how to investigate and diagnose problems or unexpected behavior with your load-balanced system.

CHAPTER 31 Further Resources

This chapter tells you where to go next for more information about Stingray traffic management products.

Glossary

The Glossary defines some terms used in this manual. Some of these terms are used with varying meanings in computing literature, so if in doubt you should refer to this section.

CHAPTER 2 Network Layouts

This chapter discusses the configuration of your network. It describes the hardware you will need for an effective traffic-managed server farm, and the DNS and IP address layout.

Essentials of Network Configuration

The components of a basic traffic-managed server farm are:

one or more front-end machines running Stingray traffic management software;

a number of back-end servers (such as web or mail servers).

The front-end machines must be able to receive traffic from the Internet (or where the remote clients are located). They must also be able to contact the back-end machines.

The back-end servers will usually be visible only from an internal network. The front-end machines do not need to route traffic between the Internet and the back-end machines.

Stingray traffic management software is commonly deployed on a **multi-homed machine.** One network interface card is visible to the Internet; one or more network interface cards are exposed to the internal private networks where the back-end servers reside. It is also easy to configure a traffic manager on a machine with a single network card (this is common in an evaluation or testing environment), where a traffic manager can contact both the clients and the servers.

A fully fault-tolerant set-up will contain two or more front-ends and several back-end servers. If any one machine fails, Stingray's fail-over capability ensures that requests are routed to other machines, ensuring there is no single point of failure in the system.

Note: Some product versions are restricted to just a cluster size of two traffic manager machines. Larger cluster sizes can be used with a software key upgrade.

If hardware availability is limited, fewer servers can be used. In the minimal case, it is possible to install the traffic management software and an Internet service on the same machine. This is not recommended, as it reduces the usefulness of the product and the ability to provide fault tolerance in the event of a hardware failure. It may, however, be useful for evaluation or demonstration purposes.

Stingray traffic management can be used in conjunction with a stand-alone firewall. The traffic manager machines should be visible both from the Internet and from the internal network; they should probably be put in the DMZ.

Note: CHAPTER 23 discusses the security aspects of network setup in more detail. You should read this section carefully before setting up live services.

Dedicated Management Network

When you configure a traffic manager, you can choose to nominate a single Management Traffic IP address. The traffic manager will only accept management traffic on that IP address.

Management traffic includes all access to the web-based Stingray Admin Server, Stingray Control API and any configuration or state sharing within a traffic manager cluster.

This can be used to provide a dedicated, trusted management network. Typically, each host running a traffic manager will have a dedicated network card that is connected to the management network.

Note: CHAPTER 23 discusses the security aspects of network setup in more detail. You should read this section carefully before setting up live services.

Important: Each management IP address is a single point of failure in a traffic manager cluster. If the management network were to fail, all inter-machine communication would be lost and remote configuration via the Admin server or Stingray Control API would be impossible.

For resilience, the fault-tolerance messages that each traffic manager machine sends are broadcast over all network cards. You can additionally restrict this traffic to the management network in the user interface.

Sizing Your Cluster

Back-End Servers

Stingray traffic management can support an unlimited number of back-end servers. These might include web servers, mail servers and FTP servers. They can run any software on any platform to provide the services Stingray manages.

The servers can be arranged into *pools* to serve different types of content. For instance, you can have several groups of web servers, including Windows machines running IIS™ and Unix machines running Zeus Web Server or Apache™, to serve

different types of static and dynamic web content. The traffic manager's system of pools and rules allows you to classify requests and send them to different groups of servers. Pools and rules are explained in CHAPTER 3.

The number of servers you dedicate to a certain function may vary. To serve static web pages, if requests are light, two Linux machines could be sufficient; while more complex Sun℠ JSP™ or Microsoft ASP pages might be served by larger numbers of machines.

Stingray traffic management allows you to use as many or few machines as you wish; should your requirements change, you can add or remove back-ends on the fly without disrupting your service.

Content Management on Back-Ends

If a number of back-ends serve the same site, changes in the site content must be propagated to each back-end server. This can be done in one of two ways:

Each back-end can store a local copy of the content, and the data can be synchronized regularly;

A shared file system can be used, such as an NFS file server. This stores the content, and each back-end server retrieves data when required.

To serve static web content, either content management method works well. For SMTP or POP3 servers, a shared file system will be needed because files are modified by that service.

Front-End Servers

A simple deployment of traffic managers would involve one front-end machine. This gives access to the full functionality of Stingray traffic management software, except for front-end fault tolerance. In other words, the traffic manager machine is a potential single point of failure.

For a fully fault-tolerant set-up, two or more traffic managers should be used. Their IP addresses can be arranged into a *Traffic IP Group* which provides fault tolerance if one machine should fail. Traffic IP groups are described in CHAPTER 6.

A common network deployment is to have the front-end machines on two separate networks: one to communicate with the Internet, and one with the back-ends (probably a private network). It is possible, however, to deploy front- and back-ends on just one network. You should ensure that all machines which are routable from the Internet are appropriately firewalled. See CHAPTER 23 for a discussion of firewalling and network security.

You may wish to increase throughput by adding extra network cards to your traffic managers. Having two network cards can also simplify deploying a traffic manager on two separate networks, although it is not necessary for this.

Stingray traffic management is highly scalable. Adding more servers or upgrading your existing hardware will increase the performance of your traffic management correspondingly. If you wish to add more traffic manager machines later, this can be done easily without disturbing your existing set-up or interrupting your service.

Note: If you are using a software version of the Stingray product family, you should ensure that the host machine is not overloaded with other applications and services. For example, running a web browser or other GUI tools on the same server will diminish the capacity of the software to manage traffic on the server.

The diagram below shows a typical traffic-managed server farm. The front-end machines have separate front- and back-end network cards as described above.

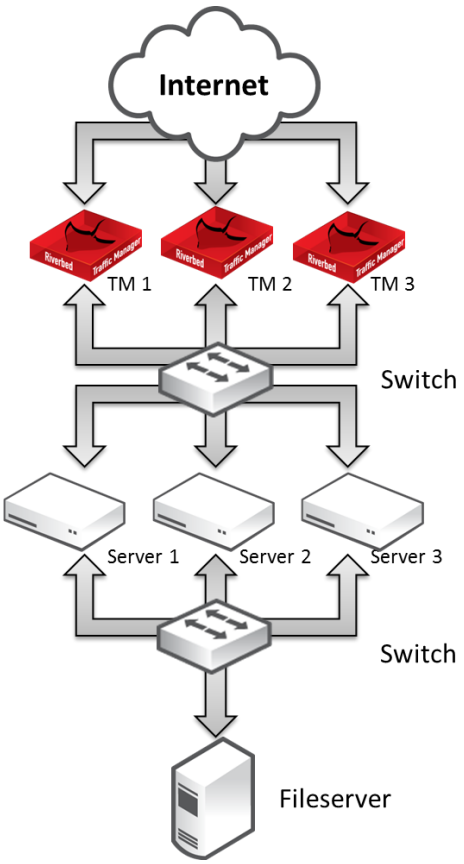


Fig. 2. Fault–tolerant network with a cluster of traffic manager units serving back-end nodes

Content served by the back-ends is held on an NFS file server. Each back-end server has two network cards, one to talk to the traffic manager machines and one to talk to this file server, to maximize throughput.

IP Transparency

Your traffic manager functions as a full application proxy, terminating network connections from remote clients and making new connections to the selected back-end servers. It does not use the packet-orientated NAT-based load balancing methods that simple layer-4 load balancers use.

With this architecture, the back-end server views the client's request as originating from the traffic manager machine, not from the remote client. This can be a disadvantage if the back-end server performs access control based on the client's IP address, or if the server wishes to log the remote IP address. This can often be worked around by performing the access control or logging functions on the traffic manager itself, or by making use of the `X-Cluster-Client-Ip` header that the traffic manager inserts into every HTTP connection to identify the client's IP address.

In situations where these workarounds are not appropriate, the traffic manager can spoof the source IP address of the server-side connection so that it appears to originate from the client's remote IP address. This capability is known as IP transparency.

IP Transparency can be used selectively. For example, if the traffic manager was balancing traffic for a web farm and a mail farm, you may wish that SMTP traffic is 'IP transparent', but not require that the web traffic is transparent.

Transparency is enabled on a per-pool basis; you can configure a web pool that is not transparent, and an SMTP pool which is transparent. The web pool and the SMTP pool can balance traffic onto the same back-end nodes, or different nodes.

Note: Stingray Traffic Manager version 9.0 does not support IP transparency for IPv6 back-ends or clients.

The IP Transparency functionality is available on all versions of the Stingray virtual appliance or cloud instance. **It is not included by default** with the Stingray traffic management software variant. It can be downloaded and installed as an additional module, and is supported on Linux kernels, version 2.6.11 and greater. Supported versions may change with future releases. Please refer to the IP Transparency documentation at http://community.riverbed.com/ for more information.

Routing configuration

Each server node that receives transparent traffic must route its responses back through the traffic manager that sent it. This is achieved by configuring the default route on the node to be one of the back-end IP addresses of the traffic manager. Please refer to your operating system documentation for details on configuring the default route.

When the server node replies to a request, it will address its response to the remote client IP. Provided the routing is configured correctly, the traffic manager system will intercept this traffic, terminate the connection and process it as normal. The traffic manager system will then forward the (possibly modified) response back to the client down the client connect.

It is normally appropriate to configure the traffic manager system to forward all other packets that are not addressed to it. This will allow the system to function as a local router, so that servers which use it as their default route can then contact other systems on nearby or remote networks. The system will need to NAT any packets that it forwards from back-end nodes on private networks.

- Please refer to your operating system documentation if you need to configure this 'IP Forwarding' capability on the system running Stingray traffic management software.
- Please refer to CHAPTER 22 if you need to configure this behavior on a Stingray virtual appliance/cloud instance.

Local routing problems

If you use IP transparency, clients on the same network as the back-end nodes will not be able to balance traffic through the traffic manager to those nodes.

This is because the back-end server nodes always attempt to reply to the source IP address of the connection. If the source IP address (the client's IP) resides on a local network, the server nodes will attempt to contact the client directly rather than routing via the traffic manager system that originated the connection. The connection will appear to hang.

In this case, it may be appropriate to segment the back-end network so that, from the server nodes' perspective, the clients appear to reside on a separate subnet, which must be reached via the default gateway (the traffic manager system). Alternatively, a TrafficScript rule could selectively set the source IP address of the server connection to an IP on the traffic manager system if the client lies on the same network as the server nodes.

IP Transparency and traffic manager clusters

IP routing is more complex in the case where a cluster of traffic managers is used, because each server node can only route back through one IP address.

Please see the *Traffic IP Addresses and Traffic IP Groups* section below for recommendations in this situation.

Traffic IP Addresses and Traffic IP Groups

In a typical network, your back-end servers will be arranged in a local network, with local IP addresses, and will not directly contactable from the Internet. The front-end traffic manager machines will have externally-available IP addresses, and will be able to connect to the back-end machines over the local network.

The front-end machines will have permanent IP addresses on each network, with the front-end addresses visible and routable from the Internet. These IP addresses are configured by the OS, and if the machine fails, the IP address is lost.

For this reason, these permanent IP addresses are not suitable to use when you publish your services. In the event of a hardware or system failure, your services would become partially or wholly unavailable.

The traffic manager's fault tolerance capability allows you to configure **Traffic IP** addresses. These IP addresses are not tied to individual machines, and the traffic manager cluster ensures that each IP address is fully available, even if some of the clustered traffic manager machines have failed.

In a typical fault-tolerant configuration, the DNS name which is used to publish your services is configured to resolve to the traffic IP addresses in your traffic manager cluster. This ensures that your services are always fully available.

The traffic IPs are arranged into a **Traffic IP group**. This group spans some or all of your traffic manager machines. The machines negotiate between them to share out the traffic IP addresses, each traffic manager then raises the IP (or IPs) allocated to it.

Setting up traffic IP groups is described in section CHAPTER 6.

If any traffic manager machine should fail, the other traffic managers in the group detect this. One of them takes over the failed machine's traffic IP address, to ensure that the service is uninterrupted.

Note: Fault Tolerance uses multicast traffic to distribute health information and to manage traffic to multi-hosted Traffic IP Addresses. The switches that your traffic managers are connected to should support IGMP snooping, and messages broadcast to the multicast address used by your traffic managers should be forwarded to all traffic managers in the cluster.

When you join a traffic manager to an existing cluster, a number of tests are conducted automatically to ascertain whether broadcast messages are correctly forwarded.

Traffic IP Address modes

Single-hosted Traffic IPs are raised on a single traffic manager in your fault tolerant cluster. If that traffic manager fails, another traffic manager will raise that IP address and start accepting traffic.

Multi-hosted Traffic IPs are raised on all of the traffic managers in your cluster, using a multicast MAC address that ensures that all incoming traffic is sent to all machines. A custom Linux kernel module is used to evenly distribute the traffic between the working traffic managers.

Enabling Multi-Hosted Traffic IPs imposes a performance hit due to the additional packet processing required by the traffic managers in your cluster. Empirical tests indicate that CPU utilization will increase by 25-30% at moderate traffic levels (10,000 requests per second), with a corresponding limit on top-end capacity.

Note: The Multi-hosted IP functionality is not included with Stingray by default. It can be downloaded and installed as an additional kernel module, and is supported on Linux kernels, version 2.6.11 and greater. Supported versions may change with future releases. Further information on this topic can be found in the **Riverbed Stingray Technical Blog** at http://blogs.riverbed.com/stingray/2009/10/multi-hosted-ip-addresses-with-zeus-software.html.

Important: In certain limited circumstances, using multi-hosted mode with particular hardware network switches can lead to a loss of availability of the hosted IP addresses. This is not a Stingray-specific issue, and there may be no related errors in the event log. If you suspect you have encountered such a problem, please refer to http://community.riverbed.com/t5/Answers/Why-can-t-users-connect-to-my-multi-hosted-IPs/m-p/16364 for more information and suggestions on how to rectify the issue.

Example Configurations

These configurations assume that you have two traffic managers in your cluster, but can be extended if your cluster contains three or more traffic managers.

Active-Passive Configuration

Suppose your web site's external DNS name maps to the IP address 162.34.64.29. You have two traffic manager machines handling traffic for a number of back-end web servers.

You can set up a single-hosted traffic IP group spanning both traffic manager machines, containing this single IP address. The traffic managers will negotiate and one of them will raise the IP address. It handles all the incoming requests. The

second traffic manager machine is available on standby. If the first machine should fail, the second machine takes over the IP address and starts to manage the traffic.

The advantage of this configuration is that you can be confident that there is sufficient resource in reserve to handle the traffic should one of the two traffic managers fail. Debugging and fault resolution is easier when only one traffic manager is handling traffic.

Active-Active Configuration – single and multi-hosted modes

In an active-active configuration, both traffic managers manage your traffic. The distribution mode (single-hosted IP or multi-hosted IP) controls how the traffic is shared between them.

With single-hosted mode, you can configure two Traffic IP Addresses in a Traffic IP Group, and configure your DNS name to map to the two addresses, such as 162.34.64.29 and 162.34.64.30. The traffic managers will negotiate to raise one traffic IP address each. A DNS server can allocate requests to each IP address in turn (***round-robin DNS***), and each traffic manager handles the requests it receives.

If one of the machines fails, the other machine will detect this. It will then raise the failed machine's traffic IP address in addition to its own, and handle all the traffic sent to either address.

With multi-hosted mode, you can continue to operate with one Traffic IP Address, simplifying your DNS and reducing the number of externally-facing IP addresses you require. The Traffic IP Address is raised on all of the traffic managers in the Traffic IP Group, and incoming traffic to that IP address is shared evenly between the traffic managers.

Active-Active with Loopback (single external address, single hosted mode)

If multi-hosted mode is not available, you can distribute traffic load to all of the traffic managers in your cluster, while still using a single external Traffic IP Address.

This configuration involves an additional 'loopback' virtual server that listens for traffic on the external Traffic IP Address, then load-balances the requests across the traffic managers using their permanent IP addresses.

First, create your primary virtual server that processes traffic and load-balances it across the back-end servers. Configure this virtual server so that it listens on internal IP addresses and ports on each traffic manager, rather than externally accessible ones. For example, the virtual server could listen on 10.100.1.1:80 and 10.100.1.2:80, where the IP addresses are internally visible only.

Then, create a second 'loopback' virtual server that listens on the single external IP address and immediately distributes traffic to the primary virtual server across the various traffic manager machines in your cluster:

- As in the active-passive example, you should set up one traffic IP address, which will be raised by one traffic manager. Any traffic coming in to this address should be processed by the simple 'loopback' virtual server listening on that traffic IP address.
- The loopback virtual server should immediately select a 'loopback' pool that contains the internal IP addresses of the traffic manager machines in the cluster; the loopback pool should use a either round-robin or least connections load balancing to evenly distribute traffic across the traffic manager machines in the cluster. It should not use a load-balancing method that is influenced by response time, as that will give very uneven request distribution.

The loopback virtual server will use little processing power. Ensure that all of the CPU-intensive processing is performed by the primary virtual server – tasks such as SSL decryption, rules, content compression etc.

This method splits the load of more intensive tasks between the two traffic managers. If either traffic manager fails, the service will continue to run (perhaps with a momentary interruption to some traffic). For example, if the traffic manager that is hosting the external Traffic IP address were to fail, the Traffic IP would be transferred to the remaining traffic manager. The loopback pool will detect that one of the nodes was unavailable and direct all traffic to the primary virtual server running on the remaining traffic manager.

The target virtual server will observe the connection originating from the loopback virtual server, not the remote client. Generally, the X-Cluster-Client-Ip header can be used to determine the correct source for the connection, but in the common case where SSL requests are forwarded by the loopback virtual server, you should use the **ssl_enhance** and **ssl_trust_magic** settings described in the *Preserving IP addresses with SSL forwarding* section in CHAPTER 13.

Multiple-Redundant (N+M) Configuration

In the cases above, two traffic manager machines are used; if one should fail, a single point of failure is introduced into the system. When running mission-critical services, a higher level of protection can be employed.

Suppose that in normal operation you wish to use *N* active traffic manager machines. You require *M* passive machines: this is the number of machines that could potentially fail without reducing the number of traffic managers in operation.

To achieve this, you would need *N+M* front-end machines running Stingray traffic management software. You can create a traffic IP group containing *N* traffic IP addresses, and spanning all *N+M* machines. This ability is also known as TrafficCluster™.

Using IP Transparency with a cluster

Using IP transparency with a cluster of traffic manager machines introduces additional complexity because each server node is configured to route to a single traffic manager IP address. However, any of the traffic manager machines in the cluster may send transparent connections to the server nodes, and the nodes must route each response back via the traffic manager that originated the connection.

Active-Passive configuration

With a single active traffic manager configuration, this can be achieved using the '**keeptogether**' setting in a traffic IP group that uses single-hosted IP addresses.

Create a traffic IP group containing two IP addresses; the front-end IP address for incoming traffic, and a back-end IP address that resides on the server side network. Select the 'keeptogether' option in the traffic IP group.

Configure each back-end server to route all traffic via the back-end IP address you configured in the traffic IP group.

With this configuration, both IP addresses will be hosted on the same traffic manager machine. If that machine were to fail, both IP addresses would be migrated to the same passive machine, making it the new active machine. The back-end servers will now route traffic back via the new active machine.

Active-Active configuration

With a multiple-active configuration, it is necessary to partition your back-end servers into two or more groups, one for each active traffic manager machine. All of the servers in each group should be configured to route traffic to the same traffic manager machine.

Your traffic manager should be configured with multiple pools, one for each server group, and TrafficScript rules can be used to select the correct pool to route traffic to based on:

The name of the traffic manager that is managing that connection, or

The local client-side IP address (if using several '**keeptogether**' traffic ip groups in single-hosted mode)

```
$hostname = sys.hostname();

if( $hostname == "ztm1" ) {
   pool.use( "ztm1_nodes" );
}
if( $hostname == "ztm2" ) {
   pool.use( "ztm2_nodes" );
}
```

IP transparency can be used selectively. For example, suppose that a traffic manager cluster is managing high volume web traffic to www.mysite.com, and low volume SMTP traffic to mail.mysite.com. Only the SMTP traffic needs to be transparent. In this case:

www.mysite.com can resolve to several IP addresses in an Active-Active TrafficCluster configuration without IP transparency.

mail.mysite.com can resolve to a single IP address using the Active-Passive '**keeptogether**' configuration described above.

Introduction to IPv6

IPv6 is a network layer protocol used in switching-packet networks. It is expected that sometime between 2010 and 2012 there will be no IPv4 addresses left for allocation. The main characteristic of IPv6 is the large amount of available addresses, as it uses 128 bits-long addresses instead of the 32 bits length provided by IPv4. It also simplifies the network management avoiding the use of complex subnetting schemes.

Some of the advantages provided by IPv6 are:

- IP security
- Mobile IP addresses
- Simplified header structure

- Address autoconfiguration
- Anycast ('one address out-of many') and mandatory multicast addresses

Main features of IPv6 in Stingray traffic management products

- Acts as a gateway for IPv6
- Can process different IP address versions at your front-end and back-end servers
- Able to work in IPv4-IPv6 mixed-networks, and in just-IPv4 networks

IPv6 unicast addresses can be used for configuring your traffic manager wherever IPv4 addresses can be used: in traffic IP addresses, when specifying nodes or the addresses a virtual server is listening on, in TrafficScript rules, etc.

Your traffic manager can also function as a gateway from IPv4 to IPv6 or vice versa and even both at the same time.

Technical restrictions

Important: Some restrictions apply when using IPv6 in the Stingray traffic management environment. Although they should not affect the normal running of the software, these restrictions must always be taken into account.

This is a list of the main restrictions regarding the use of IPv6:

- The internal communication between different traffic managers is done over the IPv4 protocol.
- Heartbeat messages only work on IPv4, as does the administration server.
- IP transparency does not work in IPv6 environments.
- When using a hostname in the configuration of a back-end node, the traffic manager will first look up the IPv4 address. If you want to use the IPv6 address of a machine where the DNS has both IPv4 and IPv6 addresses, you must enter the IPv6 address directly.
- If a host has only an IPv6 address in the DNS, that address will be used.

Tuning Duplicate Address Detection

Note: This section applies to Stingray traffic management software only. Stingray virtual appliance/cloud variants are automatically configured with Duplicate Address Detection correctly tuned.

The Duplicate Address Detection (DAD) feature of many operating systems seeks to ensure that two machines don't raise the same address simultaneously. This feature can conflict with the traffic manager's fault tolerance; when an IP is transferred from one traffic manager system to another, timing conditions may trigger DAD on the traffic manager that is raising the address. The DAD feature can be tuned as follows:

- **For FreeBSD:** set the SYSCTL parameter "`net.inet6.ip6.dad_count`" to zero. This can be achieved by editing the system configuration file /etc/sysctl.conf. Add or change the line:

  ```
  net.inet6.ip6.dad_count=0
  ```

  This change will take effect after a reboot. To adapt the setting without a system restart, issue the following command:

  ```
  # sysctl -w net.inet6.ip6.dad_count=0
  ```

  Note, however, that this setting will be lost when the system is rebooted.

- **Linux**: the sysctls are called net.ipv6.conf.default.dad_transmits and net.ipv6.conf.all.dad_transmits. Add (or change) these lines in /etc/sysctl.conf to:

  ```
  net.ipv6.conf.default.dad_transmits = 0
  ```

  and: `net.ipv6.conf.all.dad_transmits = 0`

  For immediate change, issue this command for each relevant interface (eth1 in this example), plus 'default' and 'all':

  ```
  # sysctl -w net.ipv6.conf.eth1.dad_transmits=0
  # sysctl -w net.ipv6.conf.default.dad_transmits=0
  # sysctl -w net.ipv6.conf.all.dad_transmits=0
  ```

- **Solaris 10:** if your system supports them, change the value of the following settings:

```
# ndd -set /dev/arp arp_probe_count 0
# ndd -set /dev/ip ip_dup_recovery 50
```

CHAPTER 3 Initial Configuration

This chapter explains how to set up a basic traffic-managed site.

Architecture Concepts

A traffic manager manages traffic for network services like web and application servers (HTTP, HTTPS), web services (SOAP), mail (SMTP, POP, IMAP) and other protocols such as DNS and streaming media.

A ***service*** is published as an IP address and port, and accepts traffic using the appropriate protocol.

A website is hosted at `www.mysite.com`. This DNS name resolves to the IP address 123.123.12.1; the server listens for HTTP traffic on port 80 at this address.

An ISP publishes its POP3 and SMTP mail servers as `pop.mymail.com` and `smtp.mymail.com`. Each of these servers has a DNS entry linking it to an IP address:

```
pop.mymail.com has address 202.3.45.67
smtp.mymail.com has address 202.3.45.68
```

`pop.mymail.com` listens on port 110 for POP3 traffic, while `smtp.mymail.com` listens on port 25 for SMTP traffic.

Within a traffic manager, all the traffic for a particular service is handled by a ***virtual server***. This is the interface between a traffic manager and the Internet, and is set up for a specified port and protocol; typically, it will manage all the traffic for that protocol.

When the virtual server receives a request, it assigns it to a ***pool***. This is a collection of ***nodes***, each corresponding to a back-end server and port, such as `server1.mysite.com:80`. The pool load-balances traffic across the nodes. You can set up several pools, which may have nodes in common.

To decide which pool to use for a request, the virtual server can apply a list of ***rules***. A rule inspects the incoming request, and decides what action to take with it. It can choose a pool to handle the request; close the request; or pass the request on to the

next rule in the list. If no rule makes a positive routing decision, the request is assigned to the virtual server's ***default pool***.

If a node in the pool should fail, the traffic manager's ***monitors*** detect this automatically and it stops sending requests to that node. Traffic is distributed among the other nodes in the pool with no visible disruption to the service.

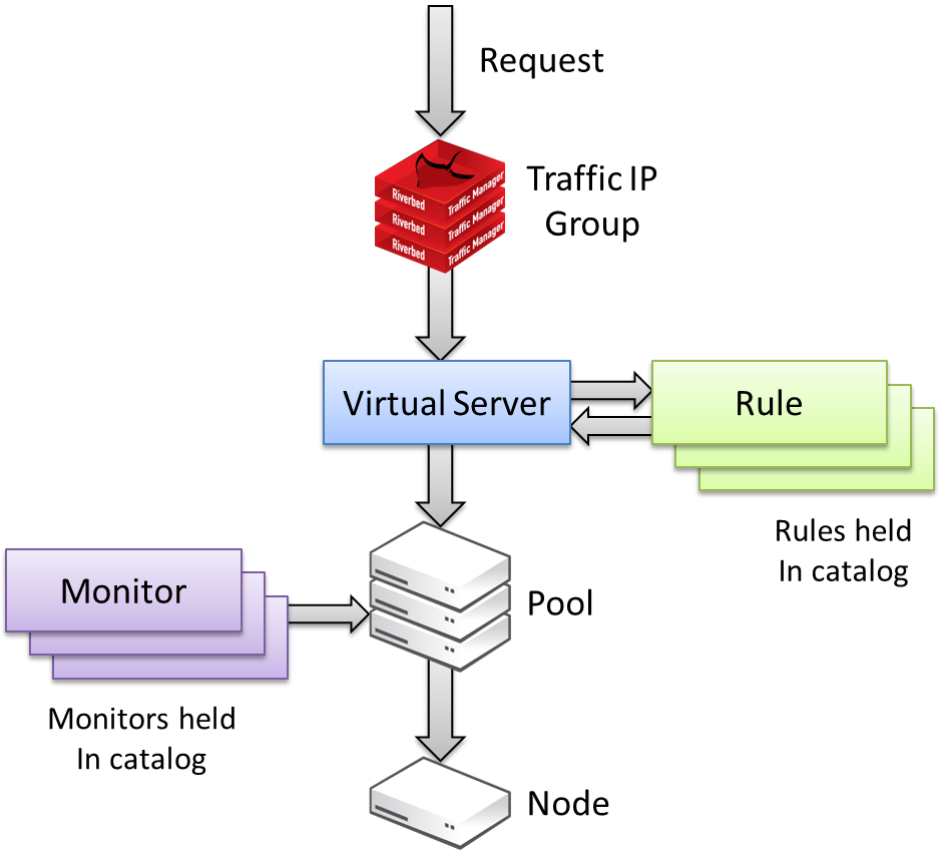


Fig. 3. A basic traffic manager configuration

A traffic manager machine can run many virtual servers, one for each service it manages. Each virtual server can use several rules and pools, and various monitors can watch the pools.

Rules and monitors are organized into ***catalogs***. The Rules Catalog, for instance, holds all the rules which have been written for the system. A rule in the catalog can be applied to any virtual server easily. Other shared items, such as SSL certificates and service protection classes, are also held in catalogs.

Two or more traffic manager machines can be arranged in a ***cluster***. Configuration is shared across all the traffic manager machines in the cluster; each machine runs the same virtual servers, using the same rules, etc.

The public traffic IP addresses used for your services can be arranged into ***traffic IP groups***. A traffic IP group spans some or all of the traffic managers in the cluster, and

these traffic managers host the group's IP addresses between them. If a traffic manager machine should fail, one of the other machines in its traffic IP group raises its IP address. This, with the pools' fail-over system, gives full fault-tolerance for both front-ends and back-ends.

Whilst a traffic manager is running hostnames are re-resolved at regular intervals, usually of no more than 5 minutes. This provides an efficient re-resolving of nodes whenever the DNS changes.

Managing your First Service

To begin managing your first service, you need to create a virtual server and a default pool. There are two ways to do this: using the **Manage a New Service** wizard or the **Configure** pages.

Browse to the address of the Admin Server home page. Log in with your username and password.

Note: If you chose not to provide a license key when you ran the 'configure' script/Startup Wizard, you can install one on the **System > Licenses** page. If no key is provided, the traffic manager will operate in Developer mode.

Using the Wizard to create a Virtual Server and Pool

Click the **"Wizards:"** drop-down menu and select the **Manage a New Service** wizard. Step through the instructions it gives.

1. Specify a name which you will use to identify the virtual server. Choose a protocol and port for the virtual server (e.g. HTTP, port 80).

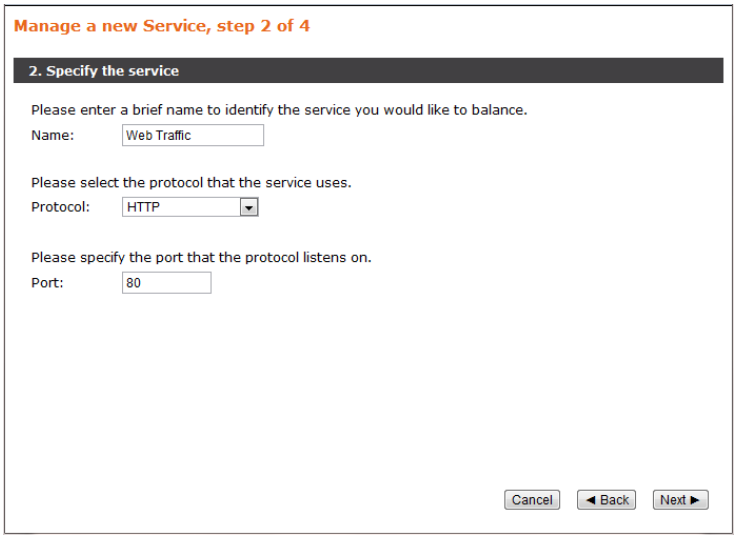


Fig. 4. Three basic parameters to define the server: name, protocol and port number

2. Create a list of back-end nodes, which will form the default pool for the virtual server. The nodes are identified by hostname and port. You can modify them later from the **Pools > Edit** page (see CHAPTER 5). You should ensure that you can serve content directly from the hostname-port combinations you specify.

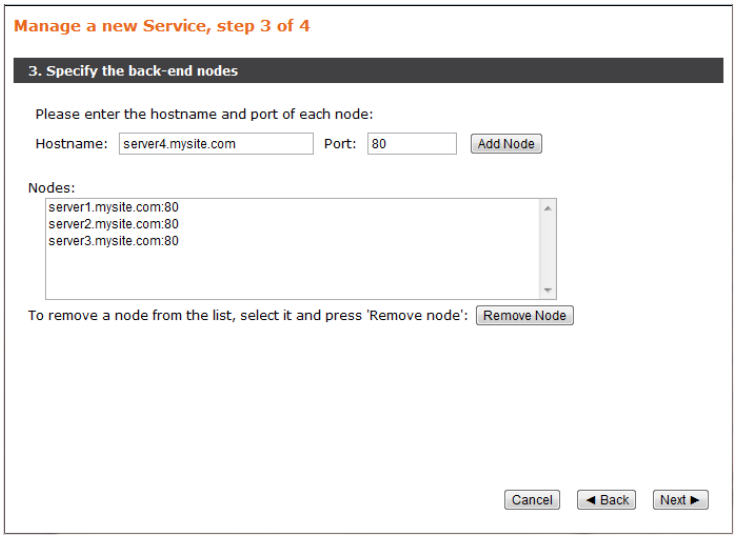


Fig. 5. Creating nodes for the server

Note: If, for evaluation purposes, you are running a traffic manager on the same machine as your web server (or other test server), you need to make sure the two services are listening on different ports.

For example, suppose you are managing traffic to a web server. The default HTTP port is 80; so you might have the web server listening on port 8080 and the traffic manager listening on port 80. This means that entering the machine name in a browser (such as server1.mysite.com) will send traffic via the traffic manager.

3. Finally, review the settings you have chosen before clicking **Finish**.

4. You can now test your traffic manager setup by browsing to the machine and port you set up for your new service.

Note: Names for virtual servers, pools, error files, etc., cannot begin with a '.', '_', or '#' and cannot or end with a '~'.

Creating a Pool and Virtual Server manually

Click the **Services** button on the top bar of the Admin interface. This takes you to the Services section, in which you can manage your virtual servers, pools, and traffic managers.

Creating a Pool

1. First you need to create a pool of back-end servers. Click the **Pools** tab and fill in the details in the **Create a New Pool** section.

   You will need to choose a name for the pool, and enter a list of nodes (unless you enable auto-scaling, in which case no nodes are required). Each node should be listed in the form `server1.mysite.com:80`, where 80 is the port `server1` is listening on. The nodes should be separated by spaces:

   ```
   s1.mysite.com:80 s2.mysite.com:8080
s3.mysite.com:80
   ```

2. Click the **Create Pool** button. You will be taken to the **Pools > Edit** page for your new pool, where you can edit its basic and more advanced settings.

Creating a Virtual Server

1. Now click the **Virtual Servers** tab to create a new virtual server. You must specify a name, a protocol, and the port the virtual server is to listen on. If you are installing the traffic manager and your server software on the same machine, note the comment about port numbers in the previous section.

   You must select a default pool for the virtual server. Requests will be assigned to this pool unless a rule specifies another pool or action.

2. Click the **Create Virtual Server** button. You are taken to the **Virtual Servers > Edit** page for your new virtual server, where you can edit basic and advanced settings. These include the IP addresses (specified by domain name or address) the virtual server listens on.

Creating a cluster

Stingray traffic manager systems are often deployed in clusters of two or more for Fault Tolerance (see CHAPTER 6) and management reasons. All of the traffic managers in a cluster share the same configuration and are managed as a single entity.

You can join traffic manager systems together to form a cluster by following the procedure below.

- If you are creating a new traffic manager cluster from scratch, you should choose one traffic manager as the first cluster member. Log in to the admin servers on the other traffic managers, and use the **Join a cluster** wizard to join each of these with the first traffic manager.
- If you want to join an existing traffic manager cluster, log in to the admin servers on the new instances and use the **Join a cluster** wizard to join each of these with the existing cluster.

Note: In a Stingray traffic management cluster, all traffic managers are considered equal. You can access the administration interface on any of the traffic managers, and configuration changes will be automatically replicated across the cluster. All of the traffic managers function together to provide fault tolerance and easy management.

Joining a cluster

Log into the admin server on one of the traffic managers and select the **Join a cluster** wizard from the drop down box next to the Help icon.

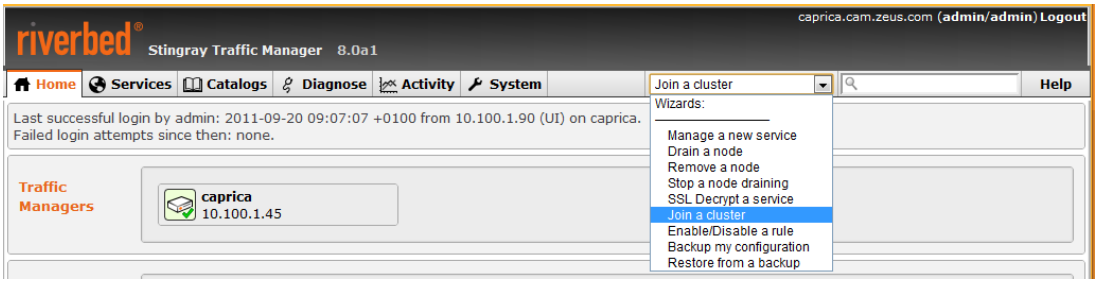


Fig. 6. Creating a cluster via the Wizard

Click the **Next** button to begin. The traffic manager will scan the network for other Stingray traffic management systems and display a list of existing machines and clusters. Select one of the other machines or clusters that you wish to join, and then click **Next**:

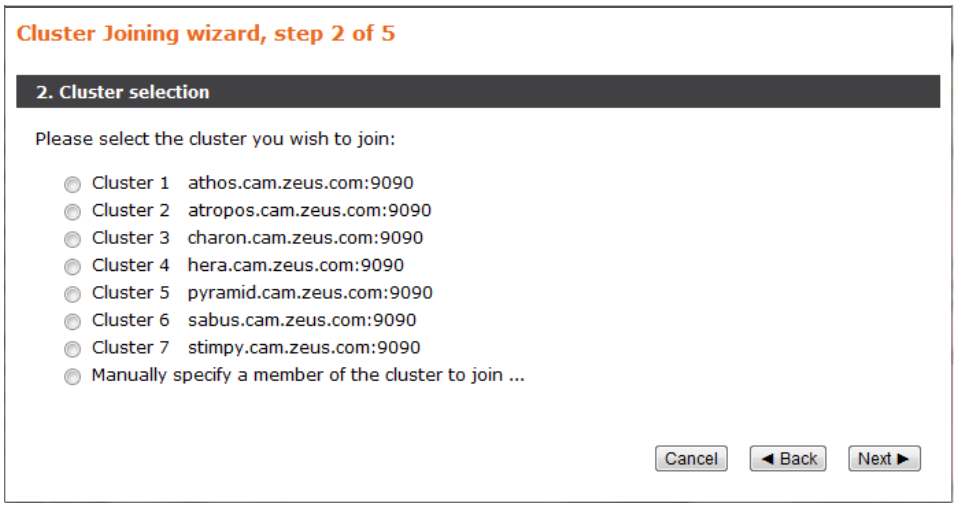


Fig. 7. Joining a cluster

You will be prompted for the admin password of the machine or cluster you have just selected. Enter it, and follow the prompts to join the cluster. The traffic manager software will reconfigure itself, and you will see a new home page showing both traffic manager systems in the Traffic Managers list.

If you wish to add further traffic managers to the new cluster you've created, log into each of the other admin GUIs, and use the join cluster wizard to join the cluster as above.

Note: When you join a traffic manager with an existing cluster, the traffic manager will take on the entire configuration that the cluster is using. This includes the administration password – all systems in a cluster use the same administration password, so the admin password for the traffic manager you are accessing may change.

If you wish to join a machine to an existing cluster, but the cluster doesn't appear in the list in the wizard, check your network configuration and cabling and check that the network between will permit multicast and broadcast packets.

Joining clusters with Traffic IP Groups

If the cluster already has one or more Traffic IP Groups, the wizard can add the new traffic manager to these Traffic IP groups so that it starts handling traffic immediately.

However, this is likely to result in a number of connections being dropped at the instant the new traffic manager is added to the Traffic IP Group, because allocations of traffic will need to be transferred to the new traffic manager. In this case, you can select to add the new traffic manager as a 'passive' member of the Traffic IP Group. It will not accept any traffic until another member of the group fails.

CHAPTER 4 Virtual Servers

A virtual server manages all the traffic for a specified port and protocol. It can apply rules to decide which pool should handle a request, and can decrypt SSL traffic if required.

Stingray traffic management solutions support a wide range of protocols. Specialized handlers are provided for complex protocols such as HTTP. RTSP and SIP, and the software can load-balance TCP connections and UDP sessions using the 'Generic' protocol versions. Details of the protocol support available can be found in CHAPTER 11.

To modify settings for a virtual server, click the **Services** button and then the **Virtual Servers** tab. Your virtual servers are listed. You can create a new virtual server here, or click the name of an existing one to access the **Virtual Servers > Edit** page for that virtual server.

How many virtual servers?

Generally, you should plan to run one Virtual Server for each distinct service you are running, i.e., each TCP or UDP port you are accepting traffic on.

For example, you should run one Virtual Server for HTTP, irrespective of how many distinct web sites you are running on port 80. This differs from the way you might configure a web server, where you configure one virtual server for each distinct web site. If you need subtly different traffic management configurations for each web site, these can be implemented using TrafficScript to control how the traffic is managed.

A typical traffic manager installation will rarely have more than 5 or 10 virtual servers configured. If you anticipate having more than 100 virtual servers, you should plan to manage your traffic manager system using the programmatic Stingray Control API (see CHAPTER 25) rather than the web-based Administration Server.

Protocols

When you configure a virtual server, you need to specify the protocol of the traffic it is handling. The protocol value is the *internal protocol* used within the traffic manager to parse and interpret the traffic. Choose the most appropriate protocol setting for the traffic you are managing.

The 'Generic Server First', 'Generic Client First' and 'Generic Streaming' protocols are simple TCP protocol handlers that process TCP connections, forwarding data between clients and servers. The three protocols differ as follows:

- When the traffic manager receives a connection on a 'Generic Server First' virtual server, it immediately runs any TrafficScript™ rules, makes a load-balancing decision and connects to a back-end server. It then relays data bi-directionally between the client and the server.

  This selection is appropriate for protocols where the server application writes a 'handshake' or 'banner' message when the client application connects, before the client writes any data to the server.

- When the traffic manager receives a connection on a 'Generic Client First' virtual server, it does not process the connection until some data has been received from the client. The traffic manager then runs TrafficScript™ rules, makes the load-balancing decision, connects to the selected back-end server and writes the data previously received from the client. The traffic manager then relays data bi-directionally between the client and the server.

  This selection is appropriate for protocols where the server expects the client to initiate the dialog by writing data first.

- The 'Generic Streaming' protocol resembles 'Generic Server First', however is an appropriate choice where the virtual server is expected to handle non-request/response data originating from either the client or the back-end server. The traffic manager runs associated TrafficScript™ rules each time, makes a load-balancing decision and then relays data bi-directionally between the client and the server.

  This selection is appropriate for protocols where the server or client may initiate the dialog, but not automatically expect a response to any data received.

These generic protocols are the foundation that other TCP protocol handlers are built upon. The generic protocols allow you to inspect and rewrite request and response data and synchronize the communications between the client and the server, but only at a very low level.

The other TCP protocol handlers in the software are specialized for particular protocols – FTP, HTTP, etc. When you select a more specialized protocol such as HTTP:

- Additional protocol-specific options are made available in the configuration of the Virtual Server and Pool, such as keepalive settings for HTTP or security settings for FTP.
- The traffic manager parses messages in that protocol and allows you to use additional TrafficScript functions to manage the request and response data more easily; helper functions to read and set Cookies in an HTTP transaction for example.

You may use the appropriate basic 'Generic' protocol handler when you are processing a high-level protocol such as HTTP. If you do so, you will be unable to use the specialized handling for that protocol.

It also supports traffic over UDP; you can select either basic UDP (for a simple UDP request-response protocol such as DNS) or UDP streaming (where the server may send a limitless number of UDP packets in response to a request).

Please refer to CHAPTER 11 for more details on specific protocol support.

Note: If the traffic is encrypted using SSL and the traffic manager decrypts the traffic, then the protocol refers to the protocol of the decrypted traffic. For example, if the traffic manager is receiving HTTPS traffic on port 443 and uses SSL decryption to decrypt it, then the protocol in use within the traffic manager is 'HTTP'.

Applying Rules

Rules can be applied to each incoming *request*, and to each outgoing *response*. They are referred to as *request rules* and *response rules* respectively.

A virtual server can examine a request using TrafficScript request rules to choose an action to take. The rules are applied in an order you specify. Each rule examines the request, possibly modifies it, and performs one of three final actions:

- Choose a pool to handle the request;
- Close the request;
- Do nothing; the request is passed to the next rule in the list.

Each virtual server is associated with a ***default pool***. If no rule makes a positive routing decision, the request is assigned to this pool.

A rule may also selectively enable or disable features in the traffic manager for that specific connection. For example, a rule can specify that the request should use a particular session persistence class, or that the response should not be cached.

Rules are constructed using the powerful TrafficScript language. This has the capability to inspect all aspects of the incoming request, from the source and destination port and IP to the type and actual content of the traffic. TrafficScript incorporates support for XPath, a language used to query XML documents, XSLT and other XML-specific capabilities. These are often used by SOAP-based protocols employed by Web services, and enable complex data to be exchanged and understood automatically without user intervention.

All the rules you create are held in the Rules Catalog. Any virtual server can use any rule from the catalog.

To add a request rule to a virtual server, go to the **Virtual Servers > Edit** page for that virtual server and click the **Rules** link. In the 'Request Rules' section, select a new rule to add from the drop-down box and click **Add Rule**.

The rules for your virtual server are shown in a list, and will be applied in the order shown. You can move a rule up or down the list by clicking on the arrow icons.

You can disable a rule in the list to temporarily stop it from being executed, and re-enable it to make it active again.

For complex protocols which conduct a lengthy dialogue with many requests and responses in one connection, you can specify whether a request rule should be run *once* (just against the first request), or against *every* request.

CHAPTER 8 covers creating and applying rules in more detail. The TrafficScript language is documented in a separate manual, included with your distribution.

SSL Decryption

A virtual server can decrypt SSL traffic. This can be useful for two reasons:

After decryption, a rule can analyze the request's headers and contents to make an informed routing decision. Without decrypting the packets very little information is available.

Decrypting requests requires processing power. It may be more efficient if the traffic manager decrypts requests before passing them on to the nodes, reducing the load on the back-end servers.

If traffic is decrypted in order to apply rules, you may wish to re-encrypt it before sending it on to the back-ends. Re-encryption is handled by the pools (see the *SSL Encryption* section in CHAPTER 4).

To set up a virtual server to decrypt SSL traffic, go to the **Virtual Servers > Edit** page for that virtual server and click on **SSL Decryption**. You can choose whether to decrypt traffic, and which certificate from the SSL Certificates Catalog to use.

You can also choose whether to request an SSL client certificate. These serve to identify the client, and you can use them to restrict access to only those individuals you choose.

The traffic manager can also check client certificates using OCSP (Online Certificate Status Protocol). OCSP is an alternative to Certificate Revocation Lists (CRLs) and allows the traffic manager to obtain the revocation status of a client certificate.

The traffic manager's SSL capability is described in detail in CHAPTER 13.

Decrypting SSL Pass-through Traffic

Recall that the protocol value for a virtual server refers to the internal protocol that the traffic manager is managing, after performing transformations such as SSL decryption.

If the protocol value for the virtual server is set to 'SSL', this indicates that the virtual server is just forwarding SSL traffic in SSL pass-through mode. If you want to configure SSL decryption, you must first change the protocol value to the correct value for the internal protocol (e.g. HTTP). In this case, your pools are probably sending traffic to nodes which expect SSL encrypted traffic, so you will also need to configure SSL encryption in the pools.

You can use the '**SSL Decrypt a Service**' wizard to configure an SSL pass-through service to decrypt traffic in the virtual server, and re-encrypt it in the pool. This wizard is described in the *SSL Decryption Wizard* section of CHAPTER 13.

Note that only some protocols support SSL decryption. SSL decryption is not available for UDP based protocols, or for protocols that cannot be automatically wrapped with SSL such as SIP.

Service Protection classes

A ***service protection class*** is a group of settings you specify to protect your service against malicious attacks, such as Denial of Service (DoS) and Distributed Denial of Service (DDoS). You can create a service protection class in the Service Protection Catalog, and configure settings such as:

Lists of banned and trusted IPs. Connections from these IPs are never allowed and always allowed, respectively.

Limits on the number of connections from one machine or a group of machines.

A limit on the connection rate from any one IP address.

Restrictions on HTTP requests, such as whether they should be strictly RFC2396-compliant.

You can apply a service protection class to a virtual server by clicking the **Classes** link on the appropriate **Virtual Servers > Edit** page. In the Service Protection section, select a class from the list and click **Update**.

Service protection is covered in detail in CHAPTER 15.

Bandwidth Management classes

A ***bandwidth management class*** defines a bandwidth limit that virtual servers can apply to data sent to clients. For example, if several large download connections were assigned a shared limit bandwidth class of 250Kbits, these connections could not consume more than this limit.

You can apply a bandwidth management class to a virtual server by clicking the **Classes** link on the appropriate **Virtual Servers > Edit** page. In the Bandwidth Management section, select a class from the list and click **Update**. This will have the effect of limiting all the traffic the virtual server manages to the defined value in the class, according to the limit sharing type specified (per-connection, per-machine, or cluster-wide).

Bandwidth Management is covered in detail in CHAPTER 16.

Note: Bandwidth Management is not available on all Stingray traffic management configurations. If required, it can be obtained via a software or license key upgrade.

Service Level Monitoring classes

Service Level Monitoring (SLM) classes are used to monitor the level of service (response time) that end users of the service are receiving. An SLM class defines a desired response time. It also defines percentage tolerance limits; if the percentage of requests that meet the desired response time falls below these limits, an alert or log message is raised.

You can apply a service level monitoring class to a virtual server by clicking the **Classes** link on the appropriate **Virtual Servers > Edit** page. In the Service Level Monitoring section, select a class from the list and click **Update**. This will cause the virtual server to monitor and log the response times for all of the connections it manages, and raise log+ messages or alerts in the event of service level problems.

Service Level Monitoring is covered in detail in CHAPTER 18.

Note: Service Level Monitoring is not available on all Stingray traffic management configurations. If required, it can be obtained via a software or license key upgrade.

HTTP Content Caching

A traffic manager HTTP virtual server can detect commonly requested HTTP resources, and remember their content if it does not change each time it is requested. This capability is called 'Content Caching'.

When the virtual server sees subsequent requests for the same resource, it can return the content for the resource directly from the local cache, rather than forwarding the request on to a (possibly overloaded) back-end server. This capability reduces the load on the back-end servers and improves the performance and capacity of your HTTP services.

Clicking the **Content Caching** link on the **Virtual Servers > Edit** page shows you the settings for content caching. You can enable the content caching capability, and specify how long various types of content are cached for. Content caching is only available for HTTP virtual servers, and for virtual servers accepting and decrypting HTTPS traffic.

Content Caching is described in detail in CHAPTER 19.

Note: HTTP Content Caching is not available on all Stingray traffic management configurations. If required, it can be obtained via a software or license key upgrade.

Aptimizer settings

Aptimizer is Stingray's web content optimization technology. It automatically optimizes web page markup and elements, such as HTML code, images, scripts and custom style sheet, so they load faster for end users. Reducing web page load time is vital to improving end user experience.

Where possible, versions of these optimized elements are internally cached for an even quicker response to subsequent requests. These accelerated web pages are

faster to load and can also save on data traffic and server infrastructure, thanks to better bandwidth utilization.

You can enable Aptimizer for your services by clicking on the **Aptimizer settings** link on the **Virtual Servers > Edit** page. Note that this facility is only available for your HTTP virtual servers.

Aptimizer settings are covered in detail in CHAPTER 20.

Note: Aptimizer functionality is not available on all Stingray traffic management configurations. If required, it can be obtained via a software or license key upgrade.

HTTP Content Compression

The traffic manager can compress an HTTP response when it sends it to the remote client. This can reduce your bandwidth usage, and speed up the delivery of large web pages to clients with slow connections.

Not all browsers can receive compressed content; those which do specify this in the HTTP request headers. The traffic manager compresses content only for those browsers which are able to decompress it.

Some web servers are also able to compress content, so it may be more efficient to spread this work across your web servers instead of using the traffic manager for this purpose. However, enabling compression on both should not cause any problems.

Clicking the **Content Compression** link on the **Virtual Servers > Edit** page shows you settings for content compression. You can choose whether to enable compression; which MIME file types to compress; and the size of documents that should be compressed.

If you offer large files for download from your site, it may be sensible to pre-compress them rather than have a server do this each time they are requested.

Controlling Content Compression

The traffic manager can compress HTTP responses if the request contains an '`Accept-Encoding: gzip`' header. The traffic manager checks the request before and after running TrafficScript rules; if the header is present in either case, the traffic manager will compress the content.

Note: Stingray traffic management does not perform the less common (and poorly supported) 'deflate' method of compression.

For example, you can configure a rule to remove the Accept-Encoding header from a request. In this case, the back-end server will never send a compressed response

(because it never sees the header), but the traffic manager will compress the response from the back-end server if the remote client supports compression. This technique can be used to offload all compression tasks from the back-end server onto the traffic manager.

You can also control whether the traffic manager compresses content using the `http.compress.enable()` and `http.compress.disable()` TrafficScript functions.

Request Tracing

The traffic manager has the ability to display a large amount of information about a particular connection. This can be a useful tool when tracing and debugging traffic handled by a virtual server. You can enable this feature under **Services > Virtual Servers > Edit > Request Tracing**. On this page you will find two options:

`request_tracing!enabled`	When enabled, the traffic manager will collect data on internal events. This data includes timing for when a request is received and when each rule is run, but it does not include information on individual reads and writes on sockets.
`request_tracing!trace_io`	When enabled, the traffic manager will collect data on individual read and write events while processing a connection.

Using Request Tracing

With the above options enabled, the **Activity > Connections** page can be used to drill down into each individual request in detail. This is done by clicking on the magnifying glass icon next to each connection. See the *Activity > Connections* section of CHAPTER 7 for more details.

Request Logging

If required, you can write detailed logs of the requests to your virtual server. You can specify a filename and log entry format, with details such as the IPs involved, the pool and node involved, and the bytes sent and received. There are also some HTTP-

specific options such as the URL requested, the value of a specified header, and the HTTP method.

Note: Unlike other logs, request logs **are not** synchronized across your cluster. Each traffic manager maintains its own log files.

You can enable request logging for a virtual server by clicking the **Request Logging** link on the appropriate **Virtual Servers > Edit** page. Click **Update** to apply your settings.

Logs can be automatically rotated so that each log does not grow too large. You can use a `%{Time-String}t` macro in the filename; this macro expands to the current time or date.

For example, the macro `%{%Y%m%d}t` expands to a value like `20051020` (for October 20th, 2005). This will cause the log files to be automatically rotated at midnight as the value of the macro changes.

On a Stingray virtual appliance/cloud instance the log file naming and rotation is handled automatically:

- The log files are checked hourly;
- Any file that reaches 1GB in size is archived;
- Once the cumulative size all current log files reach 6GB then the largest logs are archived in decreasing size order until the cumulative total goes below 6GB;
- All remaining logs are archived daily.

Once the log directory reaches 85% of capacity the traffic manager will start to delete the oldest files from the largest directories.

Viewing Request Logs

You can view request logs in real time, using the **Activity > View Logs** page described in the *Activity > View Logs* section of CHAPTER 7.

Remote Request Logging

The traffic manager can also log requests to a remote syslog server, rather than to a file on a local disk. All traffic managers in the same cluster will log traffic to the same syslog server.

To configure remote logging, click the **Request Logging** link on the appropriate **Virtual Servers > Edit** page. Enable the **syslog** capability, specify an endpoint (IP address or host, and optional port), and select the log file format you wish to use. Click **Update** to apply your settings.

Important: Note that the syslog protocol uses UDP, which is not guaranteed to be reliable. There is a danger on a busy or lossy network or when traffic levels are particularly high that some log records may be dropped.

Controlling Request Logging

Requests can be selectively logged. The TrafficScript function `request.setLogEnabled()` can be used to enable or disable logging for an individual request; if it enables logging, the settings in the **Request Logging** page are used to determine how the request is then logged (local log file, remote syslog server etc).

Connection Management

From the **Virtual Servers > Edit** page, you can access advanced settings to manage connections between remote clients and your virtual server. These include the following:

HTTP settings	Keepalive settings, and whether the traffic manager should add an `X-Cluster-Client-Ip` header to the request.
Cookie settings	Specifies how the cookies are handled by the traffic manager.
RTSP specific settings	Allows user to define a specific range of ports to be used with RTSP (RTSP virtual servers only). Refer to the RTSP chapter for more detail.
SIP settings	Specify how the traffic manager handles SIP traffic (SIP virtual servers only). See the SIP chapter for more detail.
Location Header settings	Allows URL redirection in case a node replies with a 301 (moved permanently) or 302 (temporary redirect) status code, using an incorrect protocol, hostname or port (HTTP virtual servers only).
FTP settings	Security settings and port ranges for FTP data

	connections (FTP virtual servers only).
UDP settings	Timeouts, and the maximum number of response datagrams expected for one connection (UDP virtual servers only).
Timeout settings	Timeouts for new and established connections.
Connection Error Settings	Enable detailed logging of connection errors.
Memory limits	Maximum buffer sizes for data sent by the client and returned by the server.
Low-level settings	Enable "proxy_close" to send the client FIN to the back-end server and wait for a server response instead of closing the connection immediately. Enable "so_nagle" to improve the efficiency of the TCP connection by reducing the number of packets that have to be sent over the network.

Settings specific to a certain protocol are only shown if the virtual server is managing that protocol.

You do not normally need to change any of these settings. The default values have been chosen to give good performance on a wide range of systems.

MIME Type auto-detection

The 'MIME Type' is a component of an HTTP response that describes the type of data in the response – image, HTML page, etc – so that the client can know how to render or process it.

Some web servers and web applications do not correctly set the MIME type response header, and most client software is capable of detecting when this has occurred and auto-detecting the MIME type itself.

In certain circumstances, the traffic manager also needs to know the MIME type of a response. For example, the traffic manager will only attempt to compress certain types of HTTP response data; if the MIME type is missing or incorrectly identified, the traffic manager's content compression will not operate correctly.

In this situation, you can enable 'MIME Type auto-detection' in the **HTTP Specific Settings** part of the **Connection Management** configuration. The traffic manager

will apply a range of heuristics to deduce the MIME type of the response data; auto-detection takes place when:

- There is a response body, and
- The Content-Type header is missing, or it matches the value of **mime!default**, and
- The response is not compressed

The traffic manager will de-chunk data if necessary, if chunk-transfer encoding was used.

You should only configure MIME Type auto-detection when absolutely necessary, as it will reduce the performance of the traffic manager system.

Location Header settings

This setting provides rewriting for the most usual URL redirection responses coming from a webserver, that is, 301 (object moved permanently) and 302 (object moved temporarily).

- "location!regex": enter a regular expression to match the expected response from the node (a 301/302 redirection). You can use generic characters like (.*) to match parts of the expression.
- "location!replace": enter a regular expression using $1 to $9 to redirect the response to a different object.
- "location!rewrite": select the desired response in case the response from the node doesn't match the expression provided in location!regex.

For example, when using the following values:

- "location!regex"= (.*)www.example.com/products/(.*)
- "location!replace"= $1products.example.com/$2

when the webserver returns a response like:

```
HTTP/1.1 302 Moved Temporarily
Location:
http://www.example.com/products/hacksaw.htm
```

then it will be rewritten to:

```
HTTP/1.1 302 Moved Temporarily
Location: http://products.example.com/hacksaw.htm
```

Handling Errors

The **Connection Error Settings** section on the **Connection Management** page provides settings to enable logging of connection errors encountered by your virtual server.

Connection errors may occur for a variety of reasons:

- All of the nodes in the selected pool are unavailable
- The connection with the back-end server timed out and could not be retried (see the *Retrying failed requests* section of CHAPTER 14)
- An SSL protocol error occurred in the server-side or client-side connection
- The back-end server did not return a valid response (for example, a mal-formed HTTP response)

Client connection errors are likely to be due to the actions or behavior of clients, and are mostly not under the control of the traffic manager. Server connection errors relate to the Stingray<->backend server connection and so are likely to be affected by, and ultimately can be corrected by, Stingray and your backend server configurations.

Generally, connection errors are not logged, unless they relate to errors with the back-end servers that are detected by passive monitoring.

For debugging purposes, it can be very useful to log these errors:

- **log!client_connection_failures**: this setting enables verbose logging of client-connection errors to the error log, or another location if specified by the Event Handling settings in your traffic manager.
- **log!server_connection_errors**: this setting enabled verbose logging of server connection errors, and any internal faults in the traffic manager (such as a TrafficScript rule abort because the **max_instr** limit is exceeded).

For more information, refer to the troubleshooting documentation in CHAPTER 30 of this manual.

Returning a custom error message

If the traffic manager is unable to obtain a valid response for a request, and the client connection is still alive, it will close the client connection or (for HTTP only) return the following error to the client:

Service Unavailable

The service is temporarily unavailable. Please try again later.

This error message can be replaced, using the contents of a file uploaded to the `conf/extra` resource folder. Select the error message using the configuration setting **error_file**.

Files can be uploaded to the `conf/extra` resource folder using the **Catalog > Extra Files > Miscellaneous Files** configuration page.

Stingray is capable of processing HTTP headers within your error file, regardless of the actual file type, when you wish to set explicit directions on how the file should be handled. For example, you could set the `Content-Type` header if you wish to display a file of a particular type, such as a GIF or JPG image file. In this case, you would append the following lines to the top of your binary file:

```
Content-Type: image/gif
```

This capability can be extended to override the HTTP response code, as per the following example:

```
HTTP/1.1 200 OK
```

This would force Stingray to return the desired code in its response to the client.

Note: It is important to remember to include a blank line after any headers, before the actual content of the file.

CHAPTER 5 Pools

A pool is a logical group of back-end nodes. Each node is a combination of a server name and port, such as `server1.mysite.com:80`.

A virtual server assigns requests to a pool, which load-balances them across its nodes. Each node in the pool must be able to receive requests through the port specified, using the virtual server's protocol.

As well as load balancing, each pool has its own settings for session persistence, SSL encryption of traffic, and auto-scaling. These can all be edited via the configuration page for that pool.

To access the configuration page for a pool, click the **Services** button in the top menu bar and then click the **Pools** tab. All your pools are listed. You can create a new pool on this page, or click the name of an existing one to access the **Pools > Edit** page for that pool.

Load Balancing

The traffic manager offers a choice of load-balancing algorithms which distribute requests among the nodes in the pool. The algorithms are as follows:

Round Robin	Connections are routed to each of the back-end servers in turn.
Weighted Round Robin	As for Round Robin, but with different proportions of traffic directed to each node. The weighting for each node must be specified using the entry boxes provided.
Perceptive	Monitors the load and response times of each node, and predicts the best distribution of traffic. This optimizes response times and ensures that no one server is overloaded.
Least Connections	Chooses the back-end server which currently has the smallest number of connections.

Weighted Least Connections	Chooses the back-end server which currently has the smallest number of connections, scaled by the weight of each server. Weights can be specified in the entry boxes provided.
Fastest Response Time	Sends traffic to the back-end server currently giving the fastest response time.
Random Node	Chooses a back-end server at random.

Note: (Stingray multi-site mode only) Node weightings cannot be switched between single and multiple location-based input like other areas of the Stingray UI. Instead, if you have chosen to configure your nodes by location, the associated weightings will automatically be displayed by the locations used.

Which load balancing method is best?

'**Least Connections**' is generally the best load balancing algorithm for homogeneous traffic, where every request puts the same load on the back-end server and where every back-end server is the same performance. The majority of HTTP services fall into this situation. Even if some requests generate more load than others (for example, a database lookup compared to an image retrieval), the 'least connections' method will evenly distribute requests across the machines and if there are sufficient requests of each type, the load will be very effectively shared. Weighted Least Connections is a refinement which can be used when the servers have different capacities; servers with larger weights will receive more connections in proportion to their weights.

Least Connections is not appropriate when individual high-load requests cause significant slowdowns, and these requests are infrequent. Neither is it appropriate when the different servers have different capacities. The '**Fastest Response Time**' algorithm will send requests to the server that is performing best (responding most quickly), but it is a reactive algorithm (it only notices slowdowns after the event) so it can often overload a fast server and create a choppy performance profile.

'Perceptive' is designed to take the best features of both 'Least Connections' and 'Fastest Response'. It adapts according to the nature of the traffic and the performance of the servers; it will lean towards 'least connections' when traffic is homogeneous, and 'fastest response time' when the loads are very variable. It uses a combination of the number of current connections and recent response times to trend and predict the performance of each server.

Under this algorithm, traffic is introduced to a new server (or a server that has returned from a failed state) gently, and is progressively ramped up to full operability. When a new server is added to a pool, the algorithm tries it with a single

request, and if it receives a reply, gradually increases the number of requests it sends the new server until it is receiving the same proportion of the load as other equivalent nodes in the pool. This ramping is done in an adaptive way, dependent on the responsiveness of the server. So, for example, a new web server serving a small quantity of static content will very quickly be ramped up to full speed, whereas a Java application server that compiles JSPs the first time they are used (and so is slow to respond to begin with) will be ramped up more slowly.

'Least Connections' is simpler and more deterministic than 'Perceptive', so should be used in preference when appropriate.

Caveats with load balancing algorithms

Least Connections, Fastest Response Time and Perceptive can all have unexpected behavior at very low traffic levels. Least Connections will not distribute requests if you only ever subject it to one connection at a time; Perceptive and Fastest Response Time will tend to favor nodes with known good response times and will ignore nodes that are untested.

Load balancing metrics are not shared between traffic managers in a cluster. For example, if two traffic managers use the Round Robin algorithm to distribute requests, they will each progress through the nodes in turn, but independently.

Most load balancing metrics are shared between processes (CPU cores) when the traffic management software runs on a multi-core server. The one exception is response time information; this is not shared across cores.

If you are testing your back-end servers and want to be sure that traffic is directed to all of them (and want request distribution rather than load balancing), then use **'Round Robin'** or **'Random'** for your test traffic.

Locality Aware Request Distribution (LARD)

Perceptive, Least Connections and Fastest Response Time all use a technique called LARD (Locality Aware Request Distribution) to try and send the same request to the same back-end server. This technique takes advantage of server caching; if a server returns a particular item of content, it is likely that it will be able to serve the same content quickly again because the content will be located in an internal cache (memory or disk).

When each algorithm makes its load balancing decision, it weights the decision with information as to which node processed the same request previously:

- 'Least Connections' will choose the favored node if there are several candidates for the node with least connections, and the favored node is one of them.

- Perceptive and Response Time algorithms give a light additional weight to the favored node in their internal selection.

Locality Aware Request Distribution is a lightweight way to advise the traffic manager how to route requests to back-end nodes. If you wish to mandate that requests for the same URL are sent to the same back-end server, you should use Universal Session Persistence (keyed by the URL) (see CHAPTER 12) to do this more forcefully. Alternatively, you can gain full control over request routing using TrafficScript™ and named node persistence, forward proxy or pool selection techniques.

Session Persistence

Session persistence is the process by which all requests from the same client session are sent to the same back-end server. It can be used for any TCP or UDP protocol.

A pool serving static web content usually has no requirement for session persistence; each page or image for a particular client can be served from a different machine with no ill effects. Another pool, serving an online shopping site, may use session persistence to ensure that a user's requests are always directed to the node holding details of their shopping basket.

The traffic manager offers several methods to identify requests which belong to the same session. A variety of different cookies can be used; persistence can be based on a rule; or the client's IP address can be used to identify sessions. If incoming traffic is SSL-encrypted, the SSL session ID can be used.

You can choose what to do if a persistent session is lost. This might be due to invalid session data, or because the node handling it has failed. In this case you can choose to close the connection, have requests sent to a new node, or redirect the user to a specified URL such as an error page.

You can apply session persistence to a pool by clicking the **Session Persistence** link on the **Pools > Edit** page for that pool. Select a session persistence class and click the **Update** button.

Important: Care must be taken when using a persistence class that is already in use by another pool. You should only re-use a persistence class if the nodes in *this* pool match those in the pool already using the class. Session affinity cannot be guaranteed where a persistence class is in use by two or more pools with different nodes.

Session Persistence is described in more detail in CHAPTER 12.

Bandwidth Management

The traffic manager can apply limits on the bandwidth used by connections to the back-end nodes. These limits may be useful if the back-end nodes reside in a remote datacenter, and the bandwidth to the remote site needs to be managed.

You can apply bandwidth limits to a pool by clicking the **Bandwidth Management** link on the **Pools > Edit** page for that pool. Select a bandwidth management class and click the **Update** button.

Note: The bandwidth limit only applies to data sent from the traffic manager system to the remote server node. The traffic manager does not apply a limit on the bandwidth from the remote node back to the traffic manager. Note also that bandwidth management **does not** work with UDP-based protocols. Bandwidth Management is covered in detail in CHAPTER 16.

Note: Bandwidth Management is not available on all Stingray traffic management configurations. If required, it can be obtained via a software or license key upgrade.

Health Monitoring

If the traffic manager sends a request to a node and receives no response, it assumes that node has failed. It stops sending it requests and balances traffic across the remainder of the pool.

The traffic manager's ***Health Monitors*** can run sophisticated health checks against back-end server nodes to determine whether they are operating correctly.

When a health monitor is assigned to a pool, it periodically checks each node in the pool; if it detects a certain number of failures, the traffic manager assumes that the node or pool is unavailable. These health tests are performed in addition to the traffic manager's connection tests when it attempts to send requests to and read responses from nodes.

The health monitors provide a range of tests, from simple tests, such as pinging each node, to more sophisticated tests which check that the appropriate port is open and the node is able to serve specified data, such as your home page. A number of pre-configured monitors are available to perform specific tasks (such as verify a POP3 login), and it is possible to create custom monitors to perform any test that is required.

Monitors fall into two categories: ***per-node*** and ***pool-wide***. A per-node monitor tests the health of each node in the pool. A pool-wide monitor performs tests on one

machine which influences the health of the entire pool. For example, a mail server pool might keep its data on an NFS server, which each of your back-end servers accesses. A pool-wide monitor could test this server. If it fails, none of the back-ends can retrieve the data so the whole pool is deemed to have failed.

On the **Pools > Edit** page, click the **Health Monitors** link to assign health monitors to a pool. When you have chosen your settings, click **Update** to apply them.

Health Monitors are described in more detail in CHAPTER 14.

SSL Encryption

You may wish to encrypt data before sending it from the traffic manager to the back-ends.

On the **Pools > Edit** page, click the **SSL Settings** link to specify SSL encryption settings. When you have chosen your settings, click **Update** to apply them.

SSL configuration is covered in detail in CHAPTER 13.

Connection Management

By clicking the **Connection Management** link on the **Pools > Edit** page, you can configure settings to manage the connections between the traffic manager and the nodes. These settings include:

Whether to force a connection from the traffic manager to the nodes to originate from a particular IP address;

Whether to maintain HTTP keepalive connections to the nodes;

Maximum times to establish a connection or wait from a response from a node.

You do not normally need to change any of these settings. The default values have been chosen to give good performance on a wide range of systems.

You can specify an error message that is returned to the client if all of the nodes in the pool are unavailable and no failpool is configured. The message is taken from the contents of the error file that you configure. For instance, for an HTTP service, the error file might be an HTML page stating that the service is unavailable.

To set up an error file, upload the file using the **Extra Files** section of the **Catalog**. Then return to the pool configuration and select the file from the drop-down list.

Pool Connection Limiting

Introduction

All servers or applications use certain amounts of system resources to process a connection. These resources may include threads, file descriptors and an amount of memory. Each back-end server may be able to process a certain number of concurrent connections before handling more connections would result in sub-optimal performance. In extreme cases, the server can be seen to queue, drop or refuse new connections while the overloaded system attempts to keep up with demand.

In cases such as this, putting a hard limit on the number of concurrent connections that a server receives can keep it running at an optimal level while guaranteeing that it won't get overloaded. Pool connection limiting provides a way to limit the number of concurrent connections that can be made to a pool of nodes. Connection limiting can impact a number of other features of the traffic manager. So, it is important that you read the entirety of this section before using the connection limiter.

Pool Connection Limits

Connection limiting is configured in a pool, but is actually applied to each node within that pool. If you set a connection limit of 50 (for example), each node in that pool will receive a maximum of 50 concurrent connections. Use of the connection limiting feature assumes that all of the nodes within a pool are of the same specification and are providing the same application. If the same nodes are present in multiple pools, then the potential concurrent connections for that node will be the sum of the rate limits configured for each of those pools.

Clustered Connection Limiting

Connections to back-end machines are not shared amongst traffic managers in a cluster. In an active-passive scenario, all connections will be used by the active machine and none by the passive machine. In an active-active scenario, the connection limits will effectively double. In active-active-active they will triple, and so on.

Connection Queuing

When a node is chosen by a load balancing algorithm, if the number of connections to that node is currently at the specified limit, the connection will get stored in a queue. When the demand for connections continually exceeds the permitted connections for a long enough period of time, there is a risk (in the case of HTTP for example) that the response to the browser will time out. In order to help mitigate this risk, the TrafficScript function connection.checkLimits() can be used to detect

when a pool has started queuing connections. This information can be used (for example) to respond immediately to the client with a different page rather than simply letting the response time out. Please refer to the Stingray Traffic Manager TrafficScript Overview and Reference Guide for more details.

Considerations

Connection limiting is a complicated feature, and it interacts with several other features of the traffic manager. This section describes those interactions so that you can know what to expect when using these features together.

Queue Management and SIP

RFC 3261 - SIP: Session Initiation Protocol Section 17.2.1 defines specific behaviour in regards to what must happen when the response time of a SIP server is going to take longer than 100ms. The traffic manager has been developed to adhere to this part of the SIP specification.

"The server transaction MUST generate a 100 (Trying) response unless it knows that the TU will generate a provisional or final response within 200 ms, in which case it MAY generate a 100 (Trying) response." [http://www.faqs.org/rfcs/rfc3261.html]

Connection Limiting and Request Rate Shaping

The traffic manager's request rate shaping feature uses a queue mechanism that is different from the one used by the connection limiting feature. Because a connection limit may cause a request to get queued while being subjected to connection limiting after being dequeued from a rate shaping class, it's possible for the rate of request getting sent to a node to vary from what was specified by the rate shaping class.

Connection Limiting and Service Level Monitoring

Service level monitoring can be affected by connection limiting. This is because it will be possible for a request to get queued by connection limiting. Time spent in the connection queue by a request will get added to the response time measured by an SLM class.

Enabling Pool Connection Limiting

In order to configure a connection limit for a pool, take the following steps:

1. Click **Services > Pools > Edit > Connection Management**
2. Set the `max_connections_pernode` parameter to the desired number
3. Scroll to the bottom of the page and click the **Update** button

Connection Limit Detail

`queue_timeout`	Connections that are queued for more than `queue_timeout` seconds will be discarded.
`max_queue_size`	Where the protocol for the associated virtual server is set to HTTP, the client is sent an internal server error (HTTP response code 500). The HTML page for this internal server error can be configured using the `error_file` parameter under the associated virtual server's **Connection Error Settings** configuration page.

Tracking Connection Limits

Because connection limiting involves queuing, it is possible that connections will time out. The current activity graph can be used to track the number of connections queued and the number of connections that time out while queued:

1. Navigate to **Activity > Current Activity**
2. Click the **Change data** button
3. Un-tick any boxes that are currently ticked
4. Expand the **Pools** branch of the **Values** tree
5. Tick the boxes that correspond to **ConnsQueued** and **QueueTimeouts**
6. If you want to also check that the connection limiting is working, expand the **Nodes** branch of the **Values** tree
7. Tick the **CurrentConn** box
8. Scroll to the bottom of the page
9. In the **New Settings** field, enter a name for your new chart (i.e. **Connection Limits**)
10. Click the radio button next to **New Settings**
11. Click the **Apply** button

Testing Connection Limits

The **zeusbench** utility can be used to check whether or not the connection limits you have configured are working properly. The zeusbench utility can be found in `$ZEUSHOME/admin/bin`. Running zeusbench in concurrency mode should be able to generate enough traffic to exhaust the free connections to your back-end nodes. If (for example) you have a pool of 2 nodes and you have configured a limit of 10 connections per node, the following zeusbench command should cause connections to be queued:

```
./zeusbench -c 100 -t 60 http://<your
site>/index.html
```

zeusbench will attempt to open 100 connections to the virtual server you specify (by URL) and send as many requests down each one as fast as possible. This should result in a similar number of active connections being made to the pool under test. Keep in mind that the more nodes you have, the more concurrent connections you will need to specify when invoking the zeusbench utility.

Back-end Fault Tolerance

Stingray traffic management has a strong approach to fault tolerance. Pools can be configured in a number of ways to ensure that traffic is managed as efficiently as possible.

If a node in a pool fails, this is detected by the traffic manager's ***health monitors***. It can also be detected during load balancing, if the node does not respond to a request. No more requests are sent to that node, but are instead distributed across the other nodes in the pool. When the failed node recovers, it is brought back into service slowly until the traffic manager is satisfied that it can be relied upon.

A pool can be associated with a ***failure pool***. If every node in the original pool should fail, requests will be diverted to this failure pool.

To select a failure pool, click the **Services** button and go to the **Pools** page. Click on the name of the pool you wish to edit. In the **Basic Settings** section you can select a failure pool from the drop-down list.

A typical failure pool for an HTTP service might consist of a single "sorry server". This can rewrite all requests it receives to request a simple web page, which displays an "out of service" message.

Note: The traffic manager provides a direct way to serve a simple error page if a pool fails. Instructions to set this up, via the **Connection Management** page for that pool, are given in the *Connection Management* section above.

Priority Lists

Within a pool, you can set up a ***priority list***. This allows you to group the nodes in order of priority. You can specify the minimum number of machines you wish to receive traffic at any one time.

Suppose you have three servers, `primary1`, `primary2` and `primary3`, which you wish to use for normal traffic, and two off-site backup servers, `backup1` and `backup2`. You wish always to have at least two servers available.

The two backup servers could be set up as a failure pool for the three primary servers: however, the failure pool is only used if every node in the primary pool fails. If both `primary1` and `primary3` were to fail, only `primary2` would receive traffic.

Instead, you can set up a grouped priority list. This has two priority groups: {`primary1`, `primary2`, `primary3`} and, below it, {`backup1`, `backup2`}. You specify that a minimum of two nodes must be available at any one time.

With all the servers running, all three primary servers are in use; the backup machines do not receive traffic.

If `primary1` fails, there are still two nodes in the higher group available to take requests; these two nodes handle all the traffic.

If `primary3` now fails, only one node is left in the higher group. You have specified that two servers must be available. The traffic manager starts to send requests to *both* machines in the backup group: requests are now being handled by `primary2`, `backup1` and `backup2`.

No more priority levels are available. In the event that more nodes fail, traffic is balanced across the remaining nodes. If every node fails traffic will be passed to the pool's failure pool, if it has one.

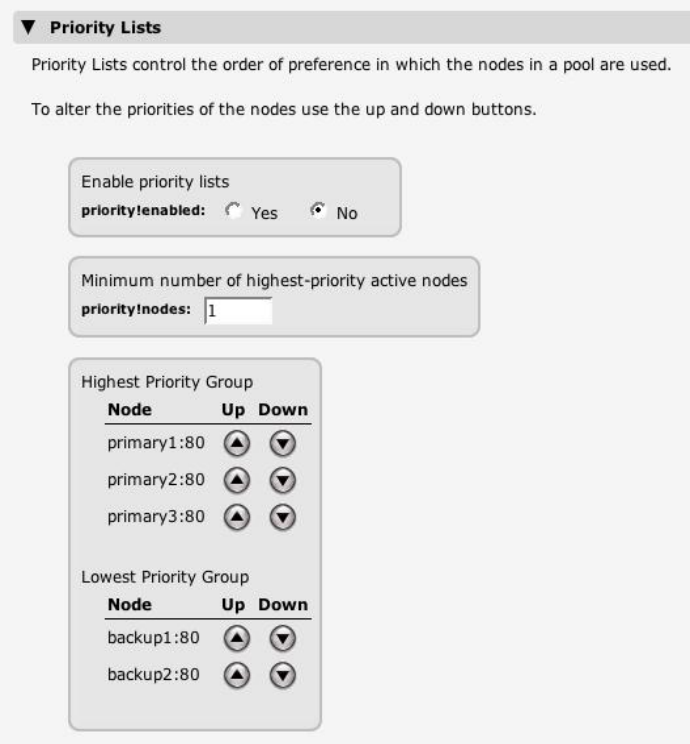


Fig. 8. Example of priority group configuration

To set up priority groups for a pool, edit that pool via the **Pools > Edit** page. Click on **Load Balancing**, and unfold the **Priority Lists** section.

The nodes in the pool are shown in a list, marked **Highest Priority Group**. To enable priority lists for this pool, click the **Yes** radio button and choose the minimum number of nodes you wish always to be available. Then use the up and down arrows beside each node to create other groups above or below the original one.

Note: The nodes in the lower priority groups are not used, until the number of nodes available in the highest group falls below the minimum you specified. Then *all* the nodes in the next group down are brought into service. The healthy servers from these top two groups are used, until enough of them fail that fewer than your specified minimum are available; at this point the *whole* of the next group is brought into use. This continues until all the priority groups are being used.

If every node in the pool should fail, requests will be directed to its failure pool, if one is configured.

Draining and Disabling Nodes

You may occasionally wish to remove one of your back-end servers from your load balanced cluster. This could be permanent, or a temporary measure to allow for maintenance or upgrade.

You can choose to either ***disable*** or ***drain*** that node so that the traffic manager will not send it any new requests:

Disable a node: This operation stops the traffic manager sending any more connections to the node. Existing connections will complete as normal. The traffic manager stops monitoring the node.

Drain a node: This operation stops the traffic manager sending any more new connections to the node, but honors established sessions. If the traffic manager receives a request and session persistence requires that the node is used, the traffic manager will use it. Please refer to the *Draining Connections* section of CHAPTER 12 for more details. The traffic manager actively monitors the node.

> You can disable or drain nodes via the **Pools > Edit** page. Edit a pool containing the node, change the status to **Disabled** or **Draining**, and click the **Update** button.

Note: This will only drain or disable the node for that particular pool. Any other pools using it will continue to send it traffic.

Disabling a node

> You should 'disable' a node if you plan to take it out of service temporarily, and you are not concerned about any sessions established with that node. If the node stops working (for example, because you shut down the server), the traffic manager will not report an error.
>
> The advantage of disabling a node rather than removing it from the configuration is that a disabled node can easily be re-instated once it is ready to be used.

Draining a node

> You should 'drain' a node if you are concerned about sessions, but bear in mind that if the node stops working, the traffic manager will report an error. If you plan to shut the node down, you should drain it first, waiting until all sessions have completed, then disable it.

To find out whether the node is still handling connections, click the **Activity** button and go to the **Draining Nodes** tab. This page shows the number of connections the node is still handling, and the time since the last connection. When the node's existing connections have expired, you can disable it, or use the **Remove a Node** wizard to remove it cleanly from your cluster.

Using the Drain a Node wizard

The **Drain a Node** wizard can be accessed from the top menu bar. You specify a node to drain, and choose whether to drain it for all services or just those you specify. You can restore the node later using the **Stop a Node Draining** wizard.

Auto-scaling

Note: Auto-scaling is not available on all Stingray traffic management configurations. If required, it can be obtained via a software or license key upgrade.

Introduction

The **application auto-scaling** option monitors the performance of a service running on a supported virtual or cloud platform. When the performance falls outside the desired service level, Stingray can then initiate an auto-scaling action, requesting that the platform deploys additional instances of the service. Stingray will automatically load balance traffic to the new instances as soon as they are available. The auto-scaling feature consists of a monitoring and decision engine, and a collection of driver scripts that interface with the relevant platform.

Auto-scaling is a property of a pool. If enabled, you do not need to provide a specific list of nodes in the pool configuration. Instead, if performance starts to degrade, additional nodes can be requisitioned automatically to provide the extra capacity required. Conversely, a pool can be scaled back to free up additional nodes when they are not required. Hence this feature can be used to dynamically react to both short bursts of traffic or long-term increases in load.

A **Service Level Monitor** (see CHAPTER 18) is used to determine when a pool needs to be auto-scaled up or down. The service level (i.e. response time) delivered by a pool is monitored closely. If the response time falls outside the desired SLA, then auto-scaling will add or remove nodes from the pool to increase or reduce resource in order to meet the SLA at the lowest cost.

How it works

As mentioned in the Introduction, the Auto-scaling mechanism consists of a *Decision Engine* and a collection of platform-dependent *Driver* scripts.

The Decision Engine

This monitors the response time from the pool, and provides scale-up/scale-down thresholds. Other parameters control the minimum and maximum number of nodes

in a pool, and the length of time the traffic manager will wait for the response time to stabilize once a scale-up or scale-down is completed.

For example, you may wish to maintain an SLA of 250ms. You can instruct the traffic manager to scale up (add nodes) if less than 50% of transactions are completed within this SLA, up to a maximum of 10 backend nodes. Alternatively, it should scale-down (remove nodes) progressively to a minimum of 1 node if more than 95% of transactions are completed.

Note: You can manually provision nodes by editing the max-nodes and min-nodes settings in the pool. If Stingray notices that there is a mismatch between the max/min and the actual number of nodes active, then it will initiate a series of scale-up or scale-down actions.

The Cloud API Driver

Stingray includes API driver scripts for Amazon EC2, Rackspace and VMware vSphere cloud environments. Before you can create an auto-scaling pool, you must first create a set of *cloud credentials* pertaining to the cloud API you wish to use. These credentials contain the information required to allow a traffic manager to communicate with the aforementioned cloud providers. The precise credentials used will depend on the cloud provider that you specify. See the *Cloud Credentials* section of CHAPTER 7 for details.

The decision engine initiates a scale-up or scale-down action by invoking the driver with the configured credentials and parameters. The driver instructs the virtualization layer to deploy or terminate a virtual machine. Once the action is complete, the driver returns the new list of nodes in the pool and the decision engine update the pool configuration.

Configuration

Important: The mechanism employed to implement auto-scaling in a pool involves allowing the traffic manager to automatically make modifications to the pool configuration file (e.g. to add or remove nodes as they are required). Where such a pool is in active use, there is a small possibility that this process of automatic updates could interfere with configuration changes made by the user through the Admin UI or SOAP interface, if they occur simultaneously. Therefore, it is recommended that users double-check that their updates have taken effect by refreshing the pool page after making a change.

Once you have created a set of credentials for your chosen cloud API, you can proceed to configure the auto-scaling properties of your selected pool. Click on the **Autoscaling** link on the **Pools > Edit** page to display the available configuration options:

Basic settings

`autoscale!enabled`	Enables or disables auto-scaling functionality. When this is enabled, nodes will be added and removed from the pool using the mechanism specified by `autoscale!external`.
`autoscale!external`	Some cloud providers have their own mechanism for performing auto-scaling. In this case, a cloud provider would monitor the performance for a group of machines (those being the machines in the pool) and will add and remove machines as needed. The cloud provider will indicate to the traffic manager that it has added or removed a machine using its API. When the traffic manager has been informed of the change, it will add or remove nodes from the auto-scaling pool accordingly. If you intend to use your cloud provider's auto-scaling mechanism, set this parameter to yes. If you intend to use the traffic manager's auto-scaling mechanism, set this to no.
`autoscale!cloudcredentials`	In order to use a cloud's API, credentials have to be provided. Sets of cloud credentials can be pre-configured under **Catalogs > Cloud Credentials** and selected from the drop-down list here.

Depending on the cloud API type of the selected `autoscale!cloudcredentials`, the set of config keys that follows will vary according to the requirements of the cloud provider:

Amazon EC2 / Rackspace Cloud settings

EC2 AMI / Image ID	This is the unique identifier of the virtual machine image from which you want to create new node instances from. The node instance will become a node in the pool when auto-scaling upwards. The unique identifier determines the operating system and the services that will be running on the nodes. In a Rackspace cloud, this is called the **imageId**[1], and in an EC2 cloud, it is called the **ImageId**.

[1]For Rackspace, the internal identifier has to be used; to get a list of the image identifiers available to a given set of cloud credentials, run the included `rackspace.pl` script manually like this:

```
$ZEUSHOME/zxtm/bin/rackspace.pl listimageids --cloudcreds=rackspace_credentials
```

where '`rackspace_credentials`' is replaced by the name of the Rackspace cloud credentials. The resultant list will provide the required identifier along with a description.

EC2 Machine Type / Flavor ID	This is the identifier that indicates the size – in terms of resources - of the virtual machine to create for instances of nodes that are added to the pool during an auto-scale event. In EC2, this is known as the **InstanceType**, Rackspace calls it the **FlavorId**[2]. Usually, more powerful machines are associated with higher costs.
Name Prefix (Rackspace only)	This name can be applied as an optional prefix to each cloud-hosted machine created by auto-scaling. A traffic manager seeing instances in the cloud starting with this name will assume those instances belong to the corresponding pool.

VMware vSphere settings

VMware Template	The built-in vSphere autoscaler script creates new Virtual Machines (VMs) by deploying them from existing VM templates. These templates are pre-configured VMs which are marked as a template upon which new VMs can be based. The path of the template on a vCenter server is usually in the form:

[2] As with the **imageId** in Rackspace, the internal identifier has to be used; to get a list of the size identifiers available to a given set of cloud credentials, run the included `rackspace.pl` script like this:

```
$ZEUSHOME/zxtm/bin/rackspace.pl listsizeids --cloudcreds=rackspace_credentials
```

where '`rackspace_credentials`' is replaced by the name of the Rackspace cloud credentials. The resultant list will provide the required identifier along with a description.

	`<DatacenterName>/vm/<TemplateName>`
Name Prefix	New nodes will be created with this name prefix on the vCenter Server. If multiple pools are auto-scaling on the same vSphere infrastructure, care must be taken to keep the name prefixes for each pool unique.
Data Center	All ESX hosts managed by the vCenter server are arranged in logical datacenters. Auto-scaling is performed on the hosts or clusters contained within this datacenter.
Data Store	The data-store that will be used by the newly created virtual machines. If left blank, the default data-store used by the datacenter will be used (depending on the vCenter configuration). **If a data-store is specified here, you should ensure the VM template uses the same data-store.**
VMware ESX Cluster	The infrastructure on a vCenter server usually has a datacenter at the top level, which contains ESX hosts or a cluster of ESX hosts. If a cluster is configured with Dynamic Resource Scheduling (DRS), vCenter decides the hosts upon which new virtual machines will be placed. If no cluster is configured, this setting should contain the ESX hostname or IP where the new VMs are to be created.

Node settings

autoscale!ipstouse	(IPs-to-use) Instances in cloud environments typically have at least two IP addresses: one on a private network for communication with other nodes in the same cloud, and one on a public network for communication with the internet in general.
autoscale!port	This defines the port to use when adding nodes to the pool. This port corresponds to the service listening on the back-end machine that gets spawned in the cloud environment.
autoscale!min_nodes	This defines the minimum number of nodes that are allowed to exist in the auto-scaling pool. Note that this number will also include nodes that are currently marked as failed. A node failure in these circumstances can lead to a degradation in response times, which in turn will result in a new node being auto-scaled in. This behavior is not guaranteed, however. An auto-scaling pool that contains failed nodes will not auto-scale beyond the number specified by autoscale!max_nodes.
autoscale!max_nodes	This defines the maximum number of nodes that are allowed to exist in the auto-scaling pool. This ensures that a pool cannot auto-scale to an enormous number by itself.

Scaling parameters

autoscale!response_time	The traffic manager monitors the response times of the nodes in an auto-scaling pool.

	The response time is measured from the time the traffic manager starts initiating a connection to the node until it receives the first byte of the response. A response time of less than `autoscale!response_time` is considered conforming and longer times are considered non-conforming. The traffic manager continuously measures the percentage of conforming connections.
`autoscale!scaledown_level`	When the percentage of conforming responses rises above this value for at least `autoscale!hysteresis` seconds, the pool is scaled down (i.e., a node is removed from the pool).
`autoscale!scaleup_level`	When the percentage of conforming responses falls below this value for at least `autoscale!hysteresis` seconds, the pool is scaled up (i.e. a node is added to the pool).
`autoscale!refractory`	After a change to the pool size has been made (be it an increase or a decrease), the traffic manager will wait `autoscale!refractory` seconds before making another change. It will take some time for the new node to have its effect on the overall response times of the pool. So to prevent too many nodes from being created (or destroyed) at a time, you may want to increase this time period.

`autoscale!hysteresis`	The amount of time in seconds that an auto-scaling condition must exist before the traffic manager instigates a change. This corresponds to (for example) the time that the pool's percentage of conforming connections must remain below `autoscale!scaleup_level` before a scale-up event occurs.

CHAPTER 6 Traffic IP Groups and Fault Tolerance

A cluster of traffic managers can distribute incoming network traffic between them, and transfer traffic shares from one to another if a traffic manager fails. Traffic distribution is configured by means of Traffic IP Groups.

The functionality referred to in this chapter may vary slightly depending on your product platform. Please refer to the appropriate **Installation and Getting Started Guide** for additional details.

Fault Tolerance

Traffic IP Addresses and Traffic IP Groups

A **Traffic IP Address** is an IP address that must remain highly available. These Traffic IP addresses are assigned to groups, called **Traffic IP Groups**.

Each Traffic IP Group is managed by some of (or all of) the traffic managers in your cluster. They cooperate and share the traffic between them, ensuring that any traffic to the Traffic IP addresses is distributed between and managed by one of the traffic managers in the cluster.

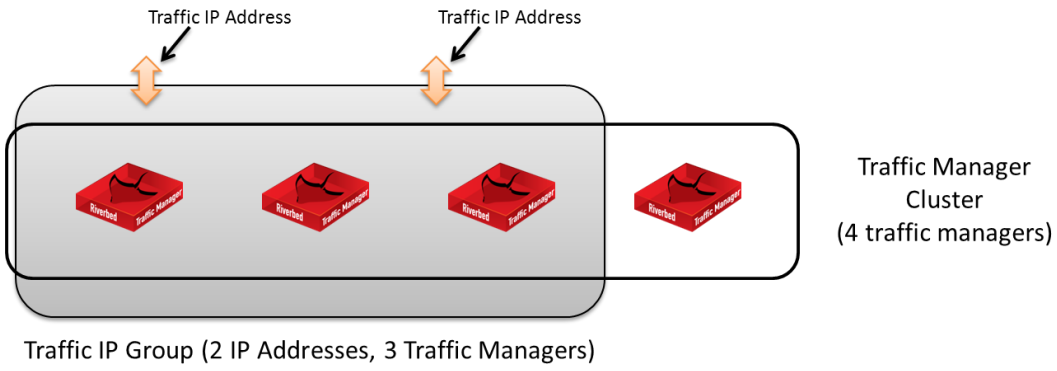


Fig. 9. Sample Traffic IP configuration.

In the illustration above, a Traffic IP Group has been configured, spanning 3 of the 4 traffic managers in the cluster. This Traffic IP Group will manage the Traffic IP addresses, and the three traffic managers will ensure that those Traffic IP Addresses are available.

For example, a web service may be published on IP address 34.56.78.90. You can ensure that the service is always available by adding that IP address to a Traffic IP group. The traffic management cluster will raise that IP address and manage all of

the traffic to it. You would typically configure the DNS entry for your service to resolve to one or more Traffic IP addresses.

Distributing traffic within a Traffic IP Group

Traffic is shared between the traffic managers using either a single-hosted or multi-hosted distribution policy:

Single-Hosted mode: Each Traffic IP Address is raised on one of the traffic managers in the group. If there are multiple IP addresses in the group, they are raised on different traffic managers, distributed as evenly as possible.

Multi-Hosted mode: Each Traffic IP Address is raised on all of the traffic managers in the group. Traffic to each IP address is evenly shared between all of the traffic managers.

Each traffic manager reports periodically to the others, and if one should fail, the remaining traffic manages would take over its share of traffic. In this way, services that depend on the Traffic IP addresses are always available.

Possible configurations of Traffic IP Groups are discussed in the *Example Configurations* section of CHAPTER 2. Details of how to create a traffic manager cluster by instructing new traffic manager machines to share their configuration with an existing traffic manager are covered in the *Creating a cluster* section of CHAPTER 3.

Creating a Traffic IP Group

To set up a Traffic IP Group, click the **Services** button and then the **Traffic IP Groups** tab. This takes you to the **Traffic IP Groups** page where you can edit your existing traffic IP groups and create new ones.

Enter a name and select the traffic managers you wish to be members of the group. Enter the traffic IPs for the group in a list separated by spaces or commas, and click **Create Traffic IP Group**.

Traffic Distribution

The IP distribution mode determines how traffic managers in the same Traffic IP Group share the traffic between them.

Traffic IP groups can be configured to be raised on one machine at a time (single-hosted mode) or all machines in the group simultaneously (multi-hosted mode).

Single-Hosted Mode

Each IP address in the group is only raised on one machine at a time. When a machine fails its Traffic IPs are moved to the other working machines in the group.

If you set the **keeptogether** option for a Traffic IP groups, then all traffic IPs will be raised on the same traffic manager. This is useful when using Traffic IP groups with IP Transparency – refer to the *Using IP Transparency with a cluster* section of CHAPTER 2 for details.

Multi-Hosted mode

Each IP address is raised on every machine at the same time. Incoming data for each IP is received by every traffic manager in the group, and each one takes responsibility for an equal portion of the incoming connections. If a machine fails, its portion is shared between the remaining active machines.

Multi-hosted IPs work by using multicast packets on your local network. The traffic manager advertises the traffic IP address using a multicast MAC address on the local subnet. This informs switches and routers that the traffic to the IP should be sent to all of the traffic managers. The MAC address used is calculated by providing a separate multicast group (represented as an IP between 239.0.0.0 and 239.255.255.255). This multicast group should be unique on your network.

Each traffic manager receives all incoming connections and makes a calculation based on the source IP address of the connection to determine whether it should accept that connection or if it should silently discard the connection (because another traffic manager in the group will accept it).

By discriminating on the source IP address alone, this method arranges that the same client will be processed by the same traffic manager (providing the client does not change IP address). This has the advantage that any session persistence data and SSL session data stored in the traffic manager will be available.

However, this can result in uneven distribution of traffic between traffic managers, particularly if your clients come from a small set of source IP addresses. This uneven distribution may impact the performance of the load-balanced service. If this is of concern, set the **consider source port** setting to instruct the traffic managers to distribute traffic based on the source IP address and port; this will share traffic much more easily across the cluster.

Note that using the source port may interfere with some session persistence methods, as there can be a short delay as the session persistence data is shared across the cluster (see the *Using Session Persistence with Multi-hosted Traffic IP Addresses* section of CHAPTER 12). Similarly, **consider source port** should not be used with any Traffic IP Addresses that are used by FTP virtual servers (see the *FTP* section of CHAPTER 11).

Note: if a traffic manager attempts to connect to a multi-hosted IP, the connection will always be picked up by the local traffic manager itself.

Passive machines

If you mark a Traffic Manager as '**passive**' then it will not raise any of the IP addresses (single-hosted mode) or handle any load (multi-hosted mode) in the Traffic IP group unless one of the non-passive traffic managers has failed.

When you add a traffic manager to an existing Traffic IP Group, some of the traffic may be transferred to the new traffic manager. Unavoidably, this will result in some dropped connections at the instant of transfer. To deal with this situation, you can add a traffic manager as 'passive'. In that case, it will not take any traffic unless one of its peers were to fail.

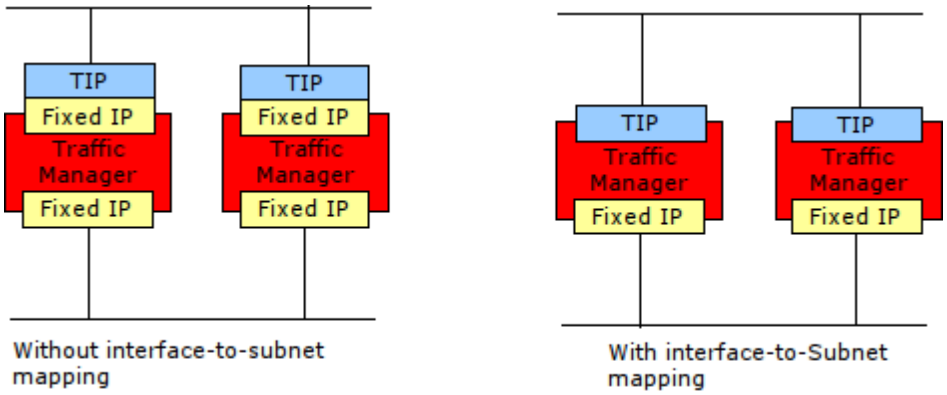
Disabling a Traffic IP Group

You can disable a Traffic IP Group by de-selecting the **enabled** checkbox. You may wish to disable a group if it is not required at present, but you do not want to delete the configuration as it may be required in the future.

When a group is disabled, none of the IP addresses in the group are raised.

Interface-to-Subnet mapping (aka Traffic IP Networks)

Prior to version 7.0 of the traffic manager, there was a pre-requisite to hosting a Traffic IP Address of having an interface that is configured with an IP address in the same subnet. This pre-requisite provides a benefit because it allows a traffic manager to automatically figure out which devices can host a Traffic IP Address (based on the routing table and "health" of the interface). The drawback to this is that it means you end up with an IP address that doesn't actually receive any traffic, but still takes up space. This can be frustrating if the network you want to receive traffic in is small (say, a /29 for example). The interface-to-subnet mapping feature gives you a way to define an explicit mapping of subnets to interfaces. When in use, you do not need to configure an interface with an IP address in a particular subnet just to host a Traffic IP Address on it.



Configuration

To configure a Traffic IP Network, take the following steps:

1. Navigate to the **Services > Traffic IP Groups** page.
2. In the **Traffic IP Networks** section, click on the **Network Settings** link.
3. In the **Add Network** field, using **CIDR notation**, enter the IP subnet that you want to host IP addresses in. For example, 192.168.50.0/24.
4. From the **Default Interface** drop-down box, select the network interface that you want this subnet to be hosted on.
5. Click the **Update** button.

After configuring a Traffic IP Subnet on the desired interface, you will see a dialogue box appear that allows you to configure which interfaces should be used for the same subnet on the other traffic managers in the cluster.

Understanding a traffic manager's fault tolerance checks

The traffic managers in a cluster periodically check that they can communicate with the network using ICMP pings through each active network interface. They then broadcast a message describing their health ('good' or 'failed').

If one of the traffic managers in a cluster fails, the other traffic managers will take over any Traffic IPs that the failed traffic manager was managing.

The Fault Tolerance checks can be configured using the **Traffic Manager Fault Tolerance** section of the **Global Settings** page, which can be found in the **System** part of the Admin Interface.

Local Health Checks

Each traffic manager checks that its network interfaces are operating correctly. It does this by:

- Periodically pinging the default gateway, to ensure that its front-end network interface is functioning.
- Periodically pinging each of the back-end nodes in the pools that are in use, to ensure that its back-end network interfaces are functioning.

The traffic manager concludes that it has failed if it cannot ping the default gateway, or if it cannot ping any of the nodes in any of the currently used pools.

Overriding Front-End Checks

You may need to override the front-end check if, for example, your gateway does not respond to ICMP pings. Edit the **flipper!frontend_check_addrs** setting in the **Global Settings** page and enter the localhost IP address (127.0.0.1) instead of the address of the default gateway.

The localhost address always responds to ICMP ping messages.

Health Broadcasts

Each traffic manager machine regularly broadcasts the results of its local health checks, whether it is healthy or not.

By default, each machine broadcasts these 'heartbeat messages' twice per second; you can configure this behavior with the **flipper!monitor_interval** setting.

You can choose to send broadcast messages by one of two methods, "**multicast"** or **"unicast"**. Select your preference with the **flipper!heartbeat_method** setting.

In most circumstances, multicast should be appropriate. However, if your network setup means that multicast messages will not be received by some traffic managers, e.g. if your traffic manager cluster spans two or more network segments, you may wish to use unicast instead.

(multicast only) The default multicast address that all traffic managers should listen on is 239.100.1.1:9090, but you can change this value using the **flipper!multicast_address** setting if network conditions require.

(unicast only) You can set the required unicast UDP port using the **flipper!unicast_port** setting.

Important: If you set the broadcast method to **unicast**, you must ensure that *State Sharing* is enabled. This is achieved by setting the **state_sync_time** configuration key to a non-zero value.

Note: The `tcpdump` program is a useful debugging tool. You can capture all heartbeat messages on your network:

```
# tcpdump –i eth0 ip multicast
```

… or just the ones issued by the traffic manager machines:

```
# tcpdump –i eth0 dst host 239.100.1.1 and dst port 9090
```

Determining the health of a cluster

Each traffic manager listens for the health messages from all of the other machines in the cluster. A traffic manager concludes that one of its peers has failed if:

- It receives a 'I have failed' health message from the peer;
- It does not receive any messages from the peer within the 5 second timeout.

You can configure the value of the timeout with the **flipper!monitor_timeout** setting.

The traffic manager concludes that a peer has recovered when it starts receiving 'I am healthy' messages from the peer.

Failover

When each traffic manager in a cluster determines that one of its peers has failed, the traffic manager may take over some or all of the traffic shares that the failed system was responsible for. The traffic distribution method determines how this is done.

Traffic IP Address transfer (Single-Hosted mode)

Each traffic manager in a cluster uses its knowledge of which machines are active to determine which Traffic IPs it should be running. The cluster uses a fully deterministic algorithm to distribute IP addresses across the machines:

Because the algorithm is deterministic, the traffic managers do not need to negotiate between themselves when one of their peers fails or recovers.

The algorithm is optimized to spread the distribution of Traffic IPs across the active traffic managers in a cluster, and to minimize the number of IP address transfers if a traffic manager fails or recovers.

When a traffic manager raises a Traffic IP address, it sends several ARP messages to inform adjacent network devices that the MAC address corresponding to the IP address may have changed. The traffic manager will send up to 10 ARP messages (tunable using the **flipper!arp_count** setting); the frequency of these messages is controlled by the **flipper!monitor_interval** setting (by default, the messages are sent at 0.5-second intervals).

Note that if a traffic manager detects that its own network connectivity has failed, it will immediately drop its Traffic IP addresses and broadcast 'I have failed' health messages to its peers. This is in anticipation of other traffic managers in the cluster raising the interfaces when they realize that the first traffic manager has failed.

Traffic IP Address transfer (multi-hosted mode)

Each traffic manager in the Traffic IP Group deterministically chooses whether or not it should handle each packet, based on the source IP of that packet (and optionally the source port, see the *Traffic Distribution* section of CHAPTER 6).

If a traffic manager fails, its share of the load is spread evenly between the remaining traffic managers. When it recovers, it takes equal shares of the load from its peers, thus ensuring that the traffic is always evenly distributed across the working machines in the Traffic IP Group.

Note: The shared IP functionality is not included with Stingray software by default. It can be downloaded and installed as an additional kernel module, and is supported on Linux kernels, version 2.6.11 and greater. Supported versions may change with future releases. Please refer to the documentation at https://support.riverbed.com/docs/stingray/index.htm for more information.

Recovering from failure

When a failed traffic manager recovers, its share of traffic is transferred back to it.

Each time traffic shares are transferred from one traffic manager to another, any connections currently in that share are dropped. This is inevitable when a transfer occurs because a traffic manager fails, but may not be desirable when a traffic manager recovers.

In this case, you can disable the **flipper!autofailback** setting in the **Global Settings > Fault Tolerance** section of the user interface. When this is disabled, a traffic manager

does not take any traffic when it recovers. Instead, the user interface displays a message indicating that the traffic manager has recovered and can take back its IPs.

When you wish to reactivate the traffic manager, go to the **Diagnose** page and select the **Reactivate this traffic manager** link.

Alternatively, you can edit each of your traffic IP groups and set the recovered traffic manager to 'passive'. Once you set it to passive in all of the groups, it will not need to take any shares of traffic; it will then reactive automatically, clearing the error state. In addition, no traffic will be lost because not traffic shares will have been transferred.

Debugging and Monitoring Fault Tolerance Activity

All state changes and IP address transfers are logged in the event logs of each relevant traffic manager. If email alerting is correctly configured (see CHAPTER 21), the traffic manager will also send an email message describing the state change and any IP address transfers that resulted to the system administrator.

For detailed debugging, you can enable the **flipper!verbose** setting. This causes each traffic manager to log every single connectivity test, broadcast message sent and broadcast message received and is useful when determining why the fault tolerance behavior is not as expected.

You can also access the **Cluster Diagnosis** page in the **Diagnose** section of the Admin Server. This page will inform you if any broadcast messages are not being received correctly, and will give you a summary of the system status if an error has occurred.

CHAPTER 7 Key features in the Stingray Administration Interface

The Home Page

The Home Page of the Stingray Administration Interface provides a dynamically-updating overview of your key configuration, traffic and status:

- The **Status Applet** displays a real-time indication of the health status of the cluster, and a real-time traffic chart and summary.

  If a problem of any kind is detected, the status light in the applet will change from green to either orange (a minor problem) or red (a major problem). Click on the status light to go directly to the Diagnosis page.
- The **Login Information Banner** provides a record of the previous successful login using these credentials. Any previous failed attempts are also shown here.
- The **Traffic Managers** section displays the status of the traffic management devices in your cluster. If any of the devices develops a fault, it will be highlighted and an error message displayed.
- The **Services** section displays a list of the services you are managing; the protocol, port, virtual server name and a list of all pools used by the service[3]. The **Stop/Start** button can be used to stop or start a service.

  If any problems are detected with a virtual server or pool, these will be highlighted.
- The **Event Log** section displays the most recent event log messages from the traffic manager systems in your cluster. Click the **Examine Logs** button to display the full event log.

The home page dynamically updates – if new event log messages are written, or if the status of a traffic manager, virtual server or pool changes, the home page will reflect this.

[3] Note: only pools that are explicitly referenced in the configuration, as a default pool, or named in a TrafficScript or RuleBuilder rule, are listed. If you select pools dynamically, using `pool.use($variable)` for example, these pools will not be displayed in this list

Note: Stingray presents a slightly different home page layout in multi-cluster management mode, including an additional section for your configured **Locations**. This is covered in detail in CHAPTER 28.

Services > Configuration Summary

With a number or virtual servers, pools and rules set up, your configuration can soon become complex. The **Config Summary** provides a single-page synopsis of the items you have configured and how they connect together.

To view the Config Summary, click the **Services** button and then the **Config Summary** tab. Your services are displayed in a table, initially grouped by port and virtual server, with each of the items used by the service displayed alongside it.

You can rearrange the table to focus in on a particular column or configuration object, and see clearly what configuration objects depend on it. For example, clicking on the **Pool** column shows you each pool name, and beside it the nodes it contains, and all the virtual servers and rules which use it.

This allows you to track exactly where each configured item is used in a complex setup.

Catalogs

The ***catalogs*** are central repositories of objects that you can use for managing traffic.

The **Rules catalog** contains TrafficScript and RuleBuilder rules..

The **Java Extensions catalog** contains Java Extensions and any supporting Java class files that the extensions require. These Java Extensions can be invoked from a TrafficScript rule.

The **Monitors catalog** stores health monitors you can use to check the correct operation of nodes in a pool.

The **SSL catalog** contains SSL resources: server and client certificates, certificate authorities and certificate revocation lists.

The **Authenticators catalog** contains definitions of remote LDAP authentication services. These services can be accessed from TrafficScript to look up information about a user and to verify their password.

The **Service Protection catalog** holds service protection classes which define the policies and security measures used to filter unwanted traffic.

The **Session Persistence catalog** contains classes which manage session persistence information for client connections.

The **Bandwidth Management catalog** contains bandwidth allocations which you can use to manage bandwidth usage.

The **Service Level Monitoring catalog** contains classes that monitor node response time and conformance to agreed levels of service.

The **Rate Shaping catalog** contains classes that can be used to queue and rate-shape requests to impose maximum request rates.

The **Cloud Credentials** catalog holds the information required to allow the traffic manager to communicate with cloud provider APIs.

The **Extra files catalog** contains files and resources that can be accessed in TrafficScript rules (using `resource.get()`), or used to provide error messages in pool configuration.

Virtual Server, Pools and Rules can each reference objects in the Catalog. If you edit an item in the catalog, the changes will be propagated to every service which uses it, making it easy to keep your configuration up to date.

To access the catalogs, click the **Catalogs** button in the top menu bar of any page. To edit or add to the items in a catalog, click the appropriate **Edit** link.

Each catalog type is described in more detail later in this document.

System > Global Settings

Stingray traffic management software provides an extensive set of configurable global settings. You can access these by clicking the **System** button and then the **Global Settings** tab. They cover aspects of the traffic manager that are not related to individual services or traffic, and are divided into the following categories:

System Settings	Define the network and system settings for each traffic manager.
Cache Settings	Control the behavior of the shared

	caches in each traffic manager.
SSL Configuration	Control the behavior of the SSL support.
SSL Hardware Support	These settings apply to any additional SSL hardware used by the traffic manager.
Logging	Control how events are logged to the event log.
Traffic Manager Fault Tolerance	Control how the traffic manager machines check for connectivity and share health information.
State Synchronization Settings	Settings to control how cluster members share their state data.
Idle Connection Settings	Settings to control how idle HTTP connections to nodes are managed.
Java Extension Settings	Settings to control how the Java Extension process is initialized and managed.
Login and Security Settings	Settings to control user login behavior (see the *Login security and behavior* section of CHAPTER 24).
Other Settings	Control other miscellaneous traffic manager settings.

Clicking on the arrow beside each category unfolds the options for that category. When you have finished making changes, click the **Update** button. You can use the **Restore Defaults** button to restore all the global settings to the system defaults.

Important: Care should be taken when modifying any of these settings. The default values have been chosen to give good performance on a wide range of systems, and to preserve the stability and security of the software. Configuring these settings incorrectly could compromise the stability and security of your system, and may impact performance.

System > Backups

You can use the traffic manager's Backup Management capability to make backups of your configuration, compare backups with each other and with the current configuration, and restore configuration from a previous backup.

You can also use the `zconf` command-line utility to perform a more fine-grained backup/import/export of individual configuration objects. See CHAPTER 27 for more details.

These solutions make it easy to copy backups from one traffic manager machine to another.

For example, you may wish to make a backup of your configuration before making a large configuration change, so that you can:

a) Compare configurations to find out exactly what you have changed;

b) Restore the earlier backup if you wish to abandon the changes you have made.

To access Backup management, click the **System** button in the top menu bar of any page and select the **Backups** tab.

Making a Backup

Use the '**Create a Backup**' form to make a backup of your configuration. This backup is stored on the local traffic manager machine.

Restoring a Backup

Click on a backup name from the table on the Backup Management page**.** Use the '**Restore Configuration**' option to replace the current configuration with the contents of the configuration backup.

A configuration backup will contain machine-specific information from the traffic manager it was taken from, such as Traffic IP Groups. Therefore, at this point you must decide whether or not to replace the current traffic manager's local machine configuration. You have two options:

- Use the machine specific configuration from the backup

  If you are restoring a backup made from *the same* traffic manager, you will simply overwrite the local configuration with the backed-up configuration. Should you have made this backup on a *different* traffic manager, you will be presented with a mapping control. Here you can

decide on the mapping for the machine configuration stored in the backup:

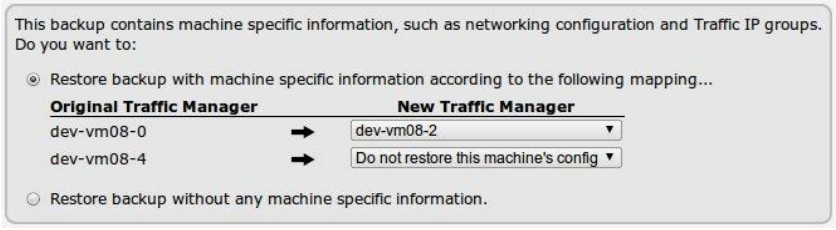


Fig. 10. In this example, the backup contains configuration for a cluster of 2 traffic managers. Using this tool, we can decide which is used for our restored configuration and which is ignored. ***Note that you can only create a one-to-one mapping between machine configurations.***

- Restore the backup without any machine specific information.

  The machine-specific configuration stored in the backup is ignored, and the current traffic manager local machine configuration retained.

Exporting a Backup

Configuration backups are stored on the local traffic manager machine. You can export a configuration backup for safekeeping.

Click on a backup name from the table on the Backup Management page**.** Use the '**Export Configuration**' option to download the configuration backup to your local machine.

Important: Store this backup securely. It contains sensitive information, including SSL certificates.

Importing a Backup

You can import a previously exported configuration backup. Use the '**Import a Backup**' option on the Backup Management page to upload a configuration backup from your local machine.

When you import a backup, it is added to the list of configuration backups on the traffic manager machine. It does not replace the current configuration. You can then restore the backup you have just uploaded to replace the current configuration if desired.

System > Backups > Partial Backups

This section provides the same backup/restore functionality of a full backup, yet allows you to tailor the contents of the backup to contain only a subset of a full configuration. It can be accessed from the link on the **System > Backups** page.

This feature provides similar functionality to the `zconf` command-line utility described in CHAPTER 27, and can be useful when trying to copy specific services to other Stingray clusters. Unlike a regular import, the configuration being imported is merged with the existing one instead of replacing it.

The page is separated into two main sections, one to handle the import and merge process, and another to enable exporting of partial backups.

Importing

This section of the page allows you to import a partial or full backup. To facilitate a partial import from the uploaded backup file, you can provide a suitable filter in the **Include Only** text box provided. The system will import any objects that match this filter, along with any objects upon which the selected objects depend. For example, the pools used by a selected virtual server will also be imported. Please refer to the **Filter formats** section below for more details of the format used.

After uploading a backup file, and optionally specifying a filter, you will be presented with a *Diff* (a list of differences) showing the changes that will be made to the system. You should review the Diff and satisfy yourself that everything is correct. Click **Apply Partial Backup** to commit the changes.

Important: Importing partial backups can lead to an inconsistent configuration if not handled correctly. Always review the *Diffs* presented when importing a partial backup. As with all backup and restore operations, it is strongly recommended that you create a full backup point before applying a partial backup.

Exporting

This section is used to download partial or full configuration backups. To facilitate a partial backup, you can set a suitable filter in the **Include Only** text box provided. The system will then export only the configuration objects that match this filter, along with their dependencies. Please refer to the **Filter formats** section below for more details of the format used.

Filter formats

The **Include Only** filter is a space separated list of entries according to one of the following formats:

```
[CONFIGURATION TYPE]          e.g. vservers
```

or

```
[CONFIGURATION TYPE]/[NAME]   e.g. vservers/Intranet
```

or

```
[CONFIGURATION FILE]          e.g. users
```

Note: When a configuration name has one or more spaces in it, you should precede each space character with a forward slash ('\'). This will ensure that the space is treated as part of the name and not a separator, e.g. "`Intranet Master Service`" would be entered as "`Intranet\ Master\ Service`".

Note: You can include the special configuration file `settings.cfg` to backup your global settings (**System > Global Settings**), and the configuration file `users` to backup your local user settings (**System > Users**).

For example, to import all virtual servers, the monitor 'Primary Database Monitor' and your global settings, you would enter the following filter:

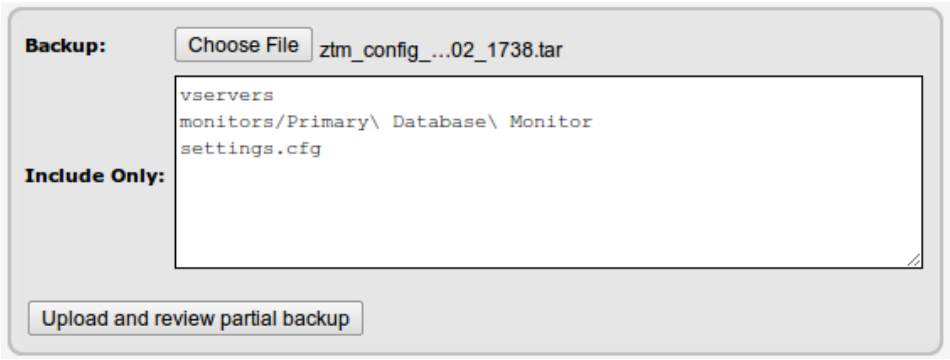


Fig. 11. The Include Only filter for a partial backup import

The following table lists the configuration identifiers you can choose to include:

Identifier	Description
actionprogs	Action Programs
actions	Alerting Actions
activitymonitor	Current Activity Graphing Data Settings
auth	Authenticators
bandwidth	Bandwidth Classes
cloudcredentials	Cloud Credentials
events	Alerting Event Types

extra	Miscellaneous Files
flipper	Traffic IP Groups
groups	User Groups
jars	Java Servlets and Related Files
licensekeys	License Keys
monitors	Monitors
persistence	Session Persistence Classes
pools	Pools
protection	Service Protection Classes
rate	Rate Shaping Classes
scripts	Monitor Scripts
services	Global Load Balancing (GLB) Services
slm	Service Level Monitoring Classes
ssl/cas	SSL Certificate Authorities
ssl/client_keys	SSL Client Certificates/Keys
ssl/dnssec_keys	DNSSEC Keys
ssl/server_keys	SSL Server Certificates/Keys
supplementarykeys	Supplementary Licence Keys
vservers	Virtual Servers

Activity Monitoring

Several visual traffic monitoring tools allow you to keep track of the current and past activity on your system. To access these tools, click the **Activity** button on the top bar of the Admin Server interface.

Activity > Current Activity

The **Current Activity** page shows you a real-time graph of the traffic activity on your system. You can customize the data extensively.

You can choose some standard sets of data to plot, traffic values such as current bandwidth, number of connections the traffic manager processes or hardware values such as free memory or spare CPU time. Much of this information can be split by virtual server, by pool, or by node. You can also dictate the size of the graph, the timescale, and choose between linear and logarithmic axes, or a pie chart. You can graph across all the machines or one at a time.

When you have chosen your settings, click **Plot Data** to view the graph. You can also download the data as a `.tsv` (tab-separated variable) file for your own analysis.

To further customize the data on the graph, click the **Change Data** button. This takes you to the **Current Activity > Edit** page where you can specify the data to plot.

In the **Global Values** section you can choose statistics to monitor and plot for the system as a whole. The **Virtual Server Values**, **Pool Values** and **Node Values** sections allow you to plot data split into individual instances of these. You can view data transfer and connection counts as well as DNS and SNMP request data, content compression and HTTP rewrites.

Service Protection Class Values section allows you to plot information about requests rejected by service protection classes. You can view data for all or a subset of your service protection classes, and split the data by the reason for rejection, such as the connection count from a single IP address being too high or the request containing disallowed content.

When you have chosen the settings to plot, you can save them to use again later. When you have finished, click **Apply**.

Monitoring Performance using SNMP

Simple Network Management Protocol (SNMP) is an open standard for network monitoring and management. It can allow you to monitor devices on a network and gather performance data.

Stingray supports multiple SNMP versions: **v1**,**v2c** and **v3**.

The traffic manager allows you to monitor performance using an SNMP tool such as HP OpenView™. You can extract information about the traffic flow for your own analysis.

Note: Stingray is also capable of sending asynchronous SNMP notifications, known as SNMP Traps. See CHAPTER 21 for more details.

To configure SNMP, click the **System** button and then the **SNMP** tab. Here you can download the relevant Stingray *MIB* (Management Information Base) for your chosen SNMP version. The MIB is a database of the information used by SNMP, and comes in two SMI (Structure of Managed Information) versions:

- **SMIv1** – for SNMPv1 clients only
- **SMIv2** – for SNMPv2c and v3 clients only

Either click on the link to view the MIB in your browser, or right-click to save the contents to a file on your computer.

Common SNMP Settings

To enable SNMP, set the `snmp!enabled` radio button to **Yes**.

You can configure common settings such as the port and bind IP address the SNMP service should listen on, and the clients that should be allowed to access the service.

The standard port for SNMP is 161, but if you wish to use an unprivileged port then 1161 is a good alternative.

SNMPv1 and SNMPv2c settings

`snmp!community`	In order to function correctly, the SNMP server must have a suitable community string. This is the equivalent of a password, and identifies authorized SNMP requests. It is not secure, as it is transmitted in the SNMP request as plain text, so you should not use a sensitive secret as the community string. The default value for many SNMP clients is "public". Since the traffic manager can restrict access to the SNMP service, the strength of this secret is not a major concern. Leaving the community string blank/empty will cause all SNMPv1 and SNMPv2c commands to be rejected.

SNMPv3 Settings

This section allows you to specify the authentication and privacy settings for accepting and responding to SNMPv3 commands.

Note: One ingredient involved in the generation of cryptographic keys from passwords is the so-called *Engine ID*. Stingray generates this value using a combination of the Zeus[4] enterprise OID (7146) and the first MAC address on the machine it is running on. The correct Engine ID value for the traffic manager you are connected to is displayed in this section.

All SNMPv3 communication is checked against the user settings that follow, and commands whose security parameters do not exactly match are rejected. (However, `snmpEngineID` discovery messages, which are never authenticated and which may be sent with an empty username, are permitted and responded to).

`snmp!username`	This is the SNMPv3 user's username, required for all SNMPv3 commands and communication. If you set this to blank/empty, all SNMPv3 commands will be rejected. The traffic manager requires and permits a **single** SNMPv3 user account to be defined. Authentication and privacy settings for this user are defined according to the fields that follow, and are based on the "User-based Security Model" described in RFC3414 (http://www.ietf.org/rfc/rfc3414.txt).
`snmp!security_level`	The SNMPv3 user's choice of whether to use authentication, and whether to use privacy. SNMPv3 *commands* must obey this choice, or they are rejected. SNMPv3 *responses* obey this choice. Selecting **Authentication** here means that commands and responses are cryptographically-signed with a key derived from `snmp!auth_password`, and this signature is included in the message. Incorrectly signed

[4] Zeus is the name of the product upon which Stingray was originally based.

	commands or responses are detected and rejected. This is intended to prevent an imposter masquerading as a trusted party, as long as the password remains secret. Additionally selecting **Privacy** here means that parts of commands and responses are encrypted using DES with a key derived from the `snmp!priv_password`. This is intended to prevent an eavesdropper from reading the content of messages, as long as the password remains secret.
`snmp!hash_alg`	This is the SNMPv3 user's choice of cryptographic hash algorithm, required for SNMPv3 commands and used for responses. Supported hash algorithms are **MD5** and **SHA-1**. See RFC 3414 for details (http://www.ietf.org/rfc/rfc3414.txt). If you set `snmp!security_level` to "**No Authentication, No Privacy**", this setting is not used.
`snmp!auth_password`	The SNMPv3 user's password for authentication. This is a shared secret; you should configure your SNMP client user account with the same authentication password. The authentication key for SNMPv3 communication is derived from this password. If you set `snmp!security_level` to "**No Authentication, No Privacy**", this setting is not used.
`snmp!priv_password`	The SNMPv3 user's password for privacy (message encryption). This is a shared secret; you should configure your SNMP client user account with the same privacy password.

	The privacy key for SNMPv3 communication is derived from this password. Re-using your authentication password here is not recommended for security reasons. If you set `snmp!security_level` to "**No Authentication, No Privacy**", or "**Authentication only**", this password is not used.

Custom Performance and Event Monitoring

The `counter.increment()` TrafficScript function is used to increment one of 10 built-in counters.

You can retrieve the values of these counters using SNMP, and you can graph them in the Activity Monitor.

These counters can be used to record custom events that you detect using TrafficScript. For example:

- If you perform authentication in a TrafficScript rule, you could count the number of authentication failures.
- If a particular class of request is particularly resource heavy, you could chart the number of times that request is made.
- If you perform security checking, for example, for a known web worm signature, you could count and chart the number of times the signature was found.

You can increase the number of built-in SNMP counters by changing the **snmp_user_counters** value located in **System>Global Settings>Other Settings**.

Activity > Historical Activity

The traffic manager records the system's activity for the past 90 days. By clicking on the **Historical Activity** tab, you can view this data in graphical form.

You can choose to display the hits per minute or the bytes processed per second, and split the results by virtual server, pool or node. Timescales to plot range from 1 hour to 30 days, and you can choose to plot the data only for certain virtual servers, pools, or nodes. You can dictate the chart size and axis type, as for the Current Activity graph.

When you have chosen your settings, click **Plot Data** to view the graph. You can also download the data as a `.tsv` (tab-separated variable) file for your own analysis.

You can change the number of days that data is stored for by changing the **statd!days** setting located in **System>Global Settings>Other Settings**.

Activity > Map

The map provides a view of the geographic origin of requests being handled by your traffic management system.

The map is designed to be interactive and tools are provided to move around and zoom into a level of detail required for your infrastructure. You can centre the map on a geographic location of your choice by holding down the left-hand mouse button and dragging the map to your required destination.

Each small dot that appears on the map represents a user who has issued a request for a particular service.

Activity > Connections

Note: This functionality described in this section is individually license-controlled. Some licenses provide a **basic** functionality mode only, with simple connection information, fewer customization options and a single filter. A **full** mode that supports all the features described here is available under the *advanced real-time analytics* license option. Contact your support provider if you wish to purchase this option.

Basic Mode

The **Connections** page gives you the ability to monitor current and recent connections being handled by the traffic manager, in date order (most recent first). This is a snapshot of connections from a point in time, where all connections after that time will not be displayed until the **Refresh Snapshot** button is pressed.

You can show and hide field columns using the button in the top left of the connections table. This displays a drop-down list of fields that the table is currently displaying. Clicking on each field in this list toggles it between being displayed or hidden.

A simple filter can be applied to the displayed data to show connections only to a specific node (the 'To' column). This is a single-field version of the filter functionality available in the **full** mode. Refer to the usage instructions described below.

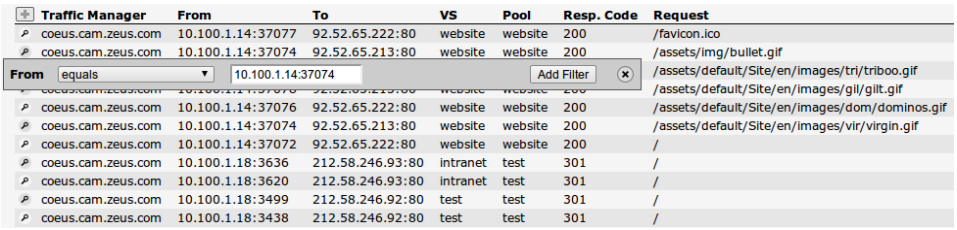
Full Mode

In this mode all of the above features are present, with additional connection detail and customization now available. You can apply a sort to the displayed data by

clicking on the column heading you wish to sort by. A small arrow next to the sorted column indicates the direction of sort. You can toggle this between ascending and descending by clicking on it.

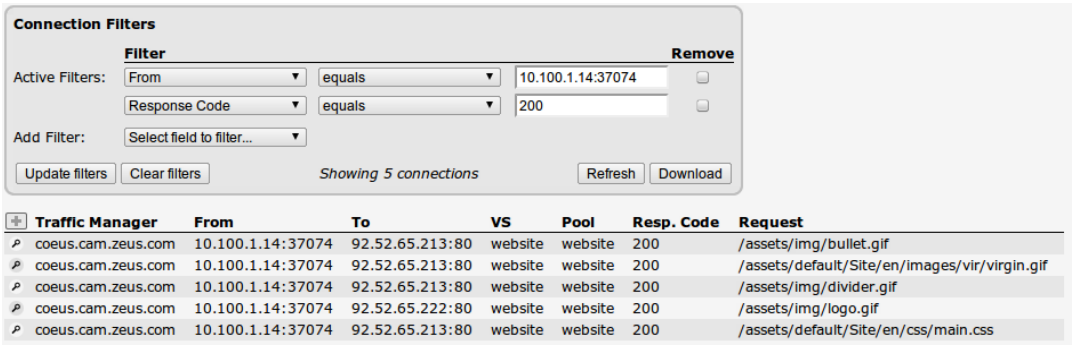
In addition to the node filter, you can now filter the connections by any field or data in the table. This can be achieved via the drop-down box in the filter section at the top of the page, or by clicking directly on one of the data values in the table. This second method produces an in-line *add filter* dialog that allows you to filter away all connections that do not share the value of the selected field.

For example, suppose you want to see requests that originate from a specific IP address and port that result in an HTTP response code of 200. First click on the IP address you wish to use in the **From** column. A filter popup will appear, populated with the selected IP:



Traffic Manager	From	To	VS	Pool	Resp. Code	Request
coeus.cam.zeus.com	10.100.1.14:37077	92.52.65.222:80	website	website	200	/favicon.ico
coeus.cam.zeus.com	10.100.1.14:37074	92.52.65.213:80	website	website	200	/assets/img/bullet.gif
coeus.cam.zeus.com	10.100.1.14:37076	92.52.65.222:80	website	website	200	/assets/default/Site/en/images/tri/triboo.gif
coeus.cam.zeus.com	10.100.1.14:37074	92.52.65.213:80	website	website	200	/assets/default/Site/en/images/gil/gilt.gif
coeus.cam.zeus.com	10.100.1.14:37076	92.52.65.222:80	website	website	200	/assets/default/Site/en/images/dom/dominos.gif
coeus.cam.zeus.com	10.100.1.14:37074	92.52.65.213:80	website	website	200	/assets/default/Site/en/images/vir/virgin.gif
coeus.cam.zeus.com	10.100.1.14:37072	92.52.65.222:80	website	website	200	/
coeus.cam.zeus.com	10.100.1.18:3636	212.58.246.93:80	intranet	test	301	/
coeus.cam.zeus.com	10.100.1.18:3620	212.58.246.93:80	intranet	test	301	/
coeus.cam.zeus.com	10.100.1.18:3499	212.58.246.92:80	test	test	301	/
coeus.cam.zeus.com	10.100.1.18:3438	212.58.246.92:80	test	test	301	/

Click the **Filter** button to apply the described logic. This filter is added to the *Active Filters* display at the top of the page. Now click on the value "200" in the **Resp. Code** column and apply a filter to this. You should now see only the desired connections.



Traffic Manager	From	To	VS	Pool	Resp. Code	Request
coeus.cam.zeus.com	10.100.1.14:37074	92.52.65.213:80	website	website	200	/assets/img/bullet.gif
coeus.cam.zeus.com	10.100.1.14:37074	92.52.65.213:80	website	website	200	/assets/default/Site/en/images/vir/virgin.gif
coeus.cam.zeus.com	10.100.1.14:37074	92.52.65.213:80	website	website	200	/assets/img/divider.gif
coeus.cam.zeus.com	10.100.1.14:37074	92.52.65.222:80	website	website	200	/assets/img/logo.gif
coeus.cam.zeus.com	10.100.1.14:37074	92.52.65.213:80	website	website	200	/assets/default/Site/en/css/main.css

Adjustments can be made to the active filters by making a change and then clicking the **Update Filters** button to redisplay the connection data. To remove a filter, tick the checkbox next to the criteria that you want to remove, and click **Update Filters**. All filters can be removed by clicking **Clear Filters**.

The **Refresh** button can be used to update the results snapshot, and the **Download** button provides a **.tsv** (tab separated value) file of the results for analysis in other applications.

Furthermore, clicking on the icon to the left of each line provides additional areas of detail for a specific connection:

Request tracing	This section gives you a timeline of internal connection processing events. This can show you how long the entire request took, how much of that time was spent processing TrafficScript rules, and other useful information. This information will recorded only if the **request_tracing!enabled** key is enabled for the virtual server that processed the connection. See the *Request Tracing* section of CHAPTER 4 for further information.
Request details	This section shows you all of the request headers that the request contained.
Response details	This section shows you all of the response headers (including those that may have been added by TrafficScript or verbose cache logging).

Note: You can increase the size of the Connections snapshot using the **System > Global Settings** page; look in the **Logging** section for the **recent_conns** key.

Activity > Draining Nodes

If you wish to remove a node from the system, for instance for maintenance or upgrade, you can set it to ***drain***. The node will continue to handle any existing sessions, but the traffic manager will not send it any more connections. The **Draining Nodes** section shows information about any nodes you are currently draining.

For each draining node, the page shows the time since the last connection, and the current number of active connections. You can update the information by clicking **Reload This Page**.

When all the active sessions have expired, it is safe to remove the node; you can do this via the **Pools > Edit** page or use the **Remove a Node** wizard.

For further information, please refer to the description of node draining in the *Draining and Disabling Nodes* section of CHAPTER 5.

Activity > View Logs

Virtual Servers may be configured to log all transactions to a Request Log. In the Virtual Server > Request Logging page (see the *Request Logging* section of CHAPTER 4), you may configure what information is logged, and where the log files are written to.

The **Activity > View Logs** page allows you to watch the request logs in real-time as records are appended to them.

Note: Stingray virtual and cloud instances are self-contained and have a dedicated, self-managed log partition. Log files are stored in this partition, archived and deleted automatically. You do not need to configure the location where request log files are stored.

Note: Log files can be viewed using the **Activity > View Logs** page, and can be downloaded using the **Activity > Download Logs** page. You can also use the Stingray Control API to download and manage log files on Stingray virtual/cloud instances.

Cloud Credentials

When using Stingray software in a cloud environment, the traffic manager might require authentication credentials in order to make API calls to the cloud provider. Primarily, pools set to use *auto-scaling* will require API credentials to enable the scaling mechanism within the cloud. This section allows you to record such credentials in a catalog object, based on the requirements of the chosen cloud provider API. Refer to the *Auto-scaling* section of CHAPTER 5 for more details on pool auto-scaling.

Stingray provides a number of built-in cloud API's upon which to base your credential set:

- VMware vSphere
- Amazon EC2
- Rackspace Cloud

Additional custom cloud APIs can be added by uploading appropriate executable scripts through the **Catalogs > Extra Files** facility. Please refer to the Stingray Community website (http://community.riverbed.com) for further details.

To create a new set of cloud credentials:

1. Click **Catalogs > Cloud Credentials**
2. Enter a name in the **Name** field
3. Select the **Cloud API** you wish to use.
4. The remaining fields differ depending on the API chosen. They typically include an ID, password/passcode, and additional authentication information.
5. Click the **Create Cloud Credentials** button.

The table below provides details for each of the fields presented when you create a new set of credentials:

`Name`	The name used to identify this set of credentials within the traffic manager's configuration.
`Cloud API`	The selected cloud provider API.
`ID / Name (cred1)`	The username or ID of the cloud provider account to be used for API calls.
`Auth Key / Password (cred2)`	The password associated with the username/ID.
`Token / Server (cred3)`	Some cloud providers also require an authentication token or additional item in order to make API calls. VMware vSphere requires a vCenter hostname/IP to accept API calls. This extra information can be specified here.
`update_interval`	The traffic manager periodically queries the cloud provider for the status of all instances running on behalf of the user. This is

	necessary in order to be up-to-date when nodes are added or removed by external systems. The `update_interval` determines how often (in seconds) these status calls are made. Note that some clouds impose a limit on the number of such calls that can be made per minute.
`change_process_timeout`	The maximum amount of time (in seconds) a change process can take. For example, when a request is made to the cloud API for node creation/destruction, this setting specifies how long to wait for the request to complete.

The Stingray Application Firewall

The Stingray Application Firewall is an enterprise-level Web Application Firewall that provides attack detection and protection for the latest generation of mission-critical web applications.

It enables centralized security monitoring, reporting and alerting and provides custom protection for your Web applications and infrastructure against external attacks.

Note: This is an optional capability of the traffic manager that is only activated through the appropriate license key upgrade. Please contact your support provider for more details.

Overview

The Stingray Application Firewall is an optional semi-standalone component of the traffic manager. Once licensed and installed/enabled in your cluster, it becomes fully integrated into the overall traffic management capability of the Stingray software. The two applications communicate with each other in order to provide a seamless service, yet each maintains a separately controllable administration interface.

The traffic manager provides the infrastructure and overall control of your web services, passing traffic to the Application Firewall when instructed to do so. Each is

administered from separate sections of the Admin UI, interlinked through buttons in the top navigation bar.

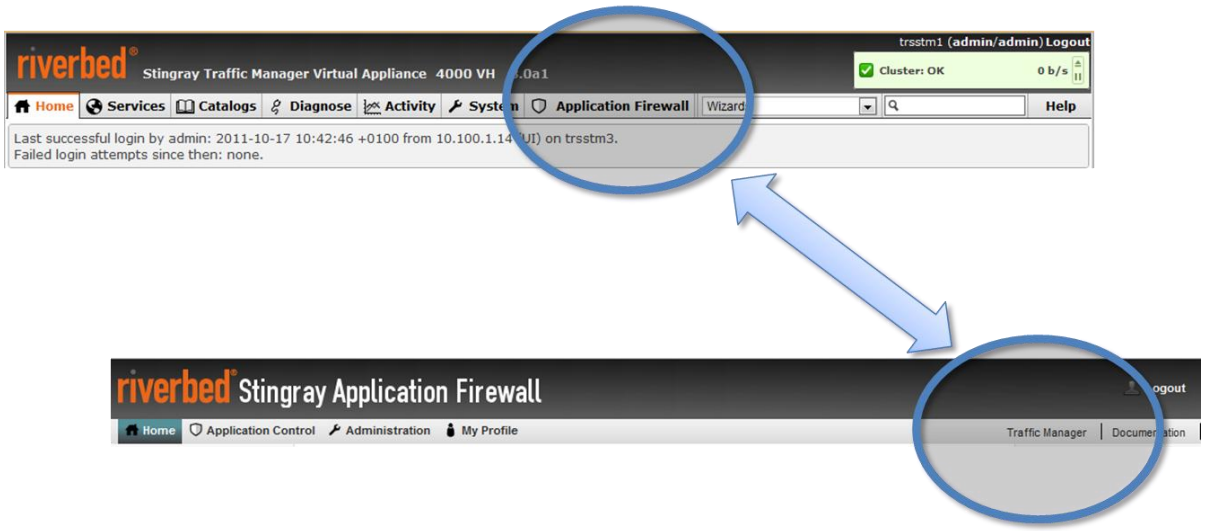


Fig. 12. Links between the main Traffic Manager and Application Firewall navigation bars

Note: By default Stingray Application Firewall is not installed or enabled, so the **Application Firewall** button shown above may not be present.

The sections that follow provide an overview of the installation procedure, features and concepts of Stingray Application Firewall. For full instructions on configuring the software, please refer to the **Stingray Application Firewall User Manual** and on-line help.

Activating Stingray Application Firewall

Important: Do not attempt to activate Stingray Application Firewall on a mixed cluster of software and non-software instances of the traffic manager. This configuration is not supported.

In order to make this functionality available, it is necessary to first enable Stingray Application Firewall through the process of purchasing and installing a suitable license key (available through your support provider). Once the key is installed, a new **System > Application Firewall** page will be made available that allows you to install, enable/disable, configure or remove the software.

The activation process from here is as follows:

1) Install Stingray Application Firewall on one traffic manager in your cluster. This can be achieved through the **System > Application Firewall** page of the Admin UI.

2) Enable the software on same traffic manager, again through the **System > Application Firewall** page.

3) Now run step 1 on each subsequent traffic manager in your cluster. Each machine will detect that there is a running Stingray Application Firewall within the cluster and will prompt you for the admin username and password in order to enable communication between the Application Firewall component on each traffic manager. The software will be enabled on each additional machine automatically.

Note: Installing Stingray Application Firewall on subsequent cluster members **before it has been enabled** on the first traffic manager may cause cluster communications errors and is to be avoided.

4) You may experience transient cluster warning messages in the traffic manager logs, status applet and on the Diagnose page. These should settle down as each machine establishes communication across the cluster.

Important: These steps impose an automatic restart of the traffic manager software. You should undertake this procedure at a time where the impact on your services is minimal.

Concepts: The Enforcer and Decider

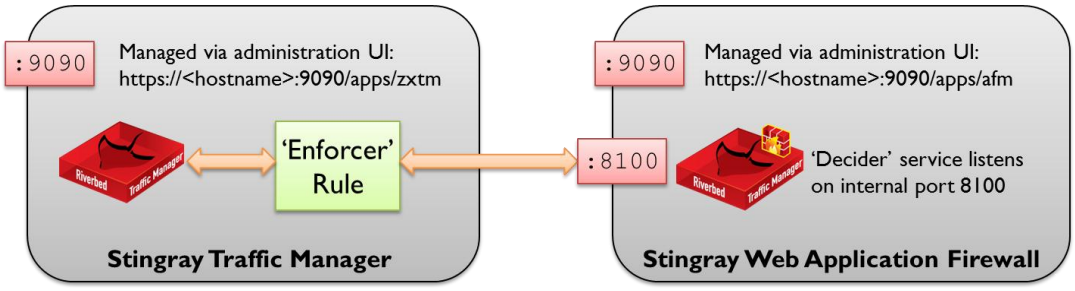


Fig. 13. Stingray Traffic Manager and the Stingray Application Firewall Decider service run on the same host. Stingray Traffic Manager's Enforcer TrafficScript rule forwards requests to the local port 8100 for inspection by the Decider. The Decider returns a 'Permit' or 'Deny' decision and the Enforcer rule applies the decision.

The **Enforcer** is a special TrafficScript Rule (entitled '*Zeus AFM Enforcer*') that integrates the Stingray traffic management component with the Application Firewall. It captures all HTTP requests and responses and forwards them to the Decider for further consideration. The decisions made by the Decider are then implemented by the Enforcer. Each request or response is accepted, modified or denied as appropriate. Further details on the Enforcer rule are provided in *The Enforcer rule* section below.

The **Decider** is a service running within Stingray Application Firewall that uses a set of rules stored in a configuration database to evaluate HTTP requests and to make decisions on the actions to be carried out.

Application Firewall features in the Traffic Manager Admin UI

Stingray Application Firewall is primarily controlled and configured via its own administration section, accessible from a link at the top of the Traffic Manager Admin UI. This link is made available ONLY after you have performed the activation procedure described in the previous section.

There are a number of additional features in the main Traffic Manager UI, discussed below:

- On the **System > Application Firewall** page you can change low-level Application Firewall settings including what network ports it uses and the number of processes it runs. This page also contains sections for enabling, disabling and uninstalling the Stingray Application Firewall software.

- A "Restart Application Firewall..." button is added to the *Software Restart* section of the **System > Traffic Managers** tab. Certain configuration changes may require a restart of Stingray Application Firewall without requiring the traffic manager itself to be restarted.

- The *Basic Settings* section of the **Services > Virtual Servers > Edit** page will contain a control to enable or disable the Application Firewall for each of your HTTP services.

- Installation of Stingray Application Firewall introduces the *Zeus AFM Enforcer* TrafficScript rule referred to in this chapter.

- The traffic manager provides status messages in its event and audit logs relating to the Application Firewall software, particularly pertaining to its current running state. Any software stops, starts and restarts will be recorded, along with specific configuration key changes and monitor errors. Stingray Application Firewall also maintains its own local event log for attack and other event information relating to the security of your services. You can view this log through the Application Firewall Admin UI. Further information can be found in the **Stingray Application Firewall User Manual**.

- A new *Stingray Application Firewall* configuration section now appears on the **Diagnose** page, which will provide more detailed descriptions of any firewall problems that occur within your cluster. Should communication between Stingray Traffic Manager and the Application Firewall be interrupted for any

reason, the **Status Applet** will show an error has occurred. Clicking on this error indicator will take you to the Stingray Application Firewall section of the Diagnose page, where you will be able to see which particular problem has occurred and determine the appropriate action to remedy the situation.

The System > Application Firewall page

There are a number of configuration options on this page that govern certain elements of Stingray Application Firewall performance and communication.

Note: During normal operation you should not need to modify any specific settings yourself, and the descriptions shown here are for information only.

The Stingray Application Firewall requires a dedicated base port that is used for communication between the Decider process and the Enforcer TrafficScript rule.

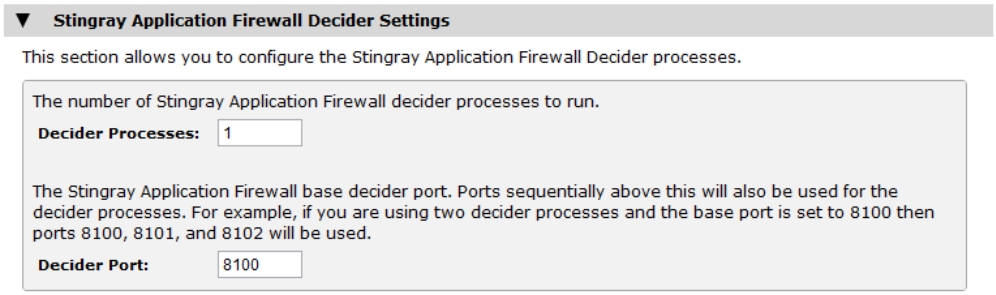


Fig. 14. The Decider settings

The default port is `8100`, however this can be modified if this port is unavailable on your system. Stingray Application Firewall is restarted automatically when changes are made.

The *Decider Processes* setting is designed to offer the option to tune the performance of Stingray Application Firewall according to the number of CPUs available on your system. Increasing this value instantiates additional Decider processes that can handle more traffic and thus improve the performance of your system. However, running more processes than there are free resources can inhibit performance and degrade the service. Reducing the number of processes in these circumstances may help increase overall system performance.

It is recommended that you set the number of decider processes to be the same as the number of CPU cores available on the machine with the fewest CPU cores in the traffic manager cluster. For example, if you have two machines in your cluster, one with four CPU cores and one with two CPU cores, you should set the number of decider processes to '2'.

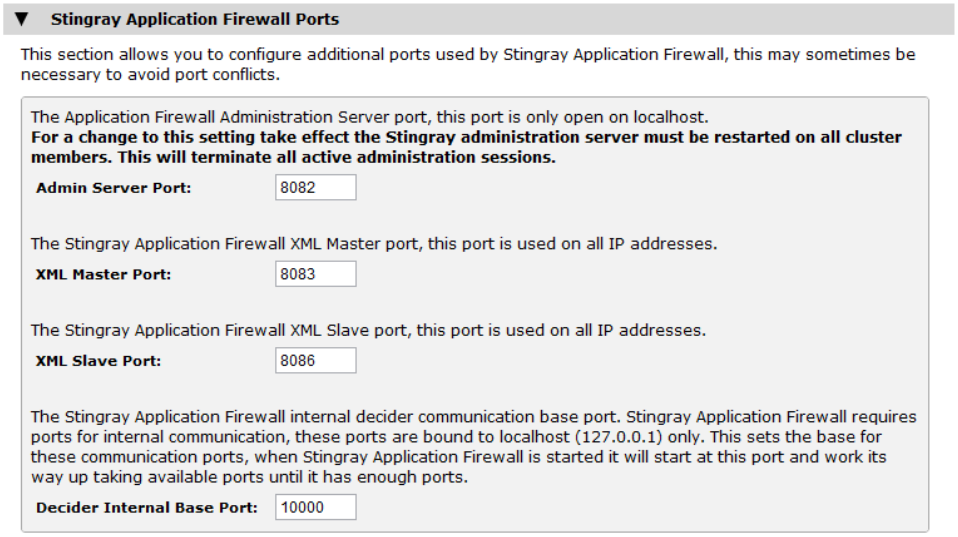


Fig. 15. Additional communications ports

Stingray Application Firewall requires a number of other ports for internal communication purposes, as shown above. Under normal circumstances, you should not need to modify the default values used. However, should there be a clash with a port number already in use you may change one or more of these settings to an alternative value here.

The Enforcer rule

Stingray Traffic Manager communicates with the Application Firewall via the TrafficScript rule system, applying certain logic to the requests and responses generated by traffic to and from your web application. A special rule entitled *Zeus AFM Enforcer* is created when the Application Firewall is enabled. It is used to instruct Stingray Traffic Manager to forward HTTP traffic to the **Decider**. The rule is attached to your Virtual Servers by enabling the special *Application Firewall* setting on the **Virtual Server > Edit** page.

Like other rules, the Enforcer can be found in the *Rules Catalog*. It has a number of configurable settings associated with it, but not all the normal rule options are available. For example, it cannot be deleted, but you can use the *Save As* functionality to make copies of it.

Note: These copies look like, and provide the same options as, the original Enforcer rule, however should be used like traditional rules. In other words, they should be assigned to virtual servers, pools and rules in the usual way. Only the original Enforcer rule is assigned using a Virtual Server's *Application Firewall* setting.

Click on the rule name, or the *Edit* link, to access its settings page. Note that the actual TrafficScript text is not immediately visible, however you can click the *Rule text* link to display it if you wish. Note that it is in a binary format and is only

partially human-readable. The top of the rule text shows a number of parameters that correspond to the settings on this page. Any modifications to these settings, listed below, are automatically updated into the rule text:

- **allowonerror**

  Set this to 1 to allow all traffic if the Decider is un-contactable. (default: 0)

- **backend_timeout**

  This is the number of seconds to wait for the Decider before timing-out. (default: 5)

- **debug**

  Set this to 1 to enable debugging mode. This should provide a more verbose error output to assist in tracking down problems. (default: 0)

- **enforcer_max_body_size**

  Reject requests with a body larger than this many bytes. (default: 65536)

Note: You should not normally need to modify these settings. Doing so may inhibit the ability of the rule to function correctly, so only proceed if you are fully aware of the consequences.

User management

User and group access between the traffic manager and Stingray Application Firewall is maintained automatically by default. A member of the **admin** group in the traffic manager UI will be created with the same level of authorization in the Application Firewall Admin UI[5], and performing usual administrative tasks will not require additional access privileges or login processing. Clicking **Logout** in either UI component logs the user out of both applications.

Important: If a user is deleted from the traffic manager, it is not automatically removed from Stingray Application Firewall. An administrator must perform this step manually.

[5] Specifically, a Stingray Application Firewall user record will not be created UNTIL the user first accesses the Application Firewall Admin UI from the traffic manager UI. If the username already exists, no new user is created and the existing user record is used instead.

Members of other groups in the traffic manager, such as **Demo** or **Monitoring**, will only gain access to the various Stingray Application Firewall UI features if the correct permission settings exists on the **System > Users > Groups > Edit Group** page. Three settings are provided in the *System* section for this purpose:

- **Application Firewall**

  Controls access to the **System > Application Firewall** page in the traffic manager UI.

- **Application Firewall > Administration**

  Controls access to the Stingray Application Firewall administration UI (through the toolbar button shown in Fig. 12 above).

- **Application Firewall > Uninstall**

  Controls access to the *Uninstall* feature on the **System > Application Firewall** page of the traffic manager UI.

Note: It is possible to override the default mechanism of automatic admin user creation by pre-defining user accounts in the Stingray Application Firewall Admin UI (perhaps with limited access or a member of some other non-admin group). A user with the same name in the traffic manager will then have only the desired access in the Application Firewall UI.

CHAPTER 8 TrafficScript Rules

This chapter describes the system of TrafficScript rules used on a traffic manager. It explains how to create and apply new rules and provides some examples.

Note: Some Stingray traffic management variants do not support TrafficScript rules or XML capabilities, or just have limited support (RuleBuilder only). Full support can be obtained via a software or license key upgrade.

The Stingray Community website contains a large amount of examples and documentation describing how to use TrafficScript rules to solve a range of network and application problems. It is located at http://community.riverbed.com.

Overview

You can customize the traffic manager using your own traffic management rules. These rules are created using a scripting language called ***TrafficScript***.

TrafficScript rules are executed by the traffic manager whenever it receives a new connection or network request, and whenever it receives a response from a node. The rules can inspect the incoming and outgoing data in the connection, and other aspects such as the remote client address.

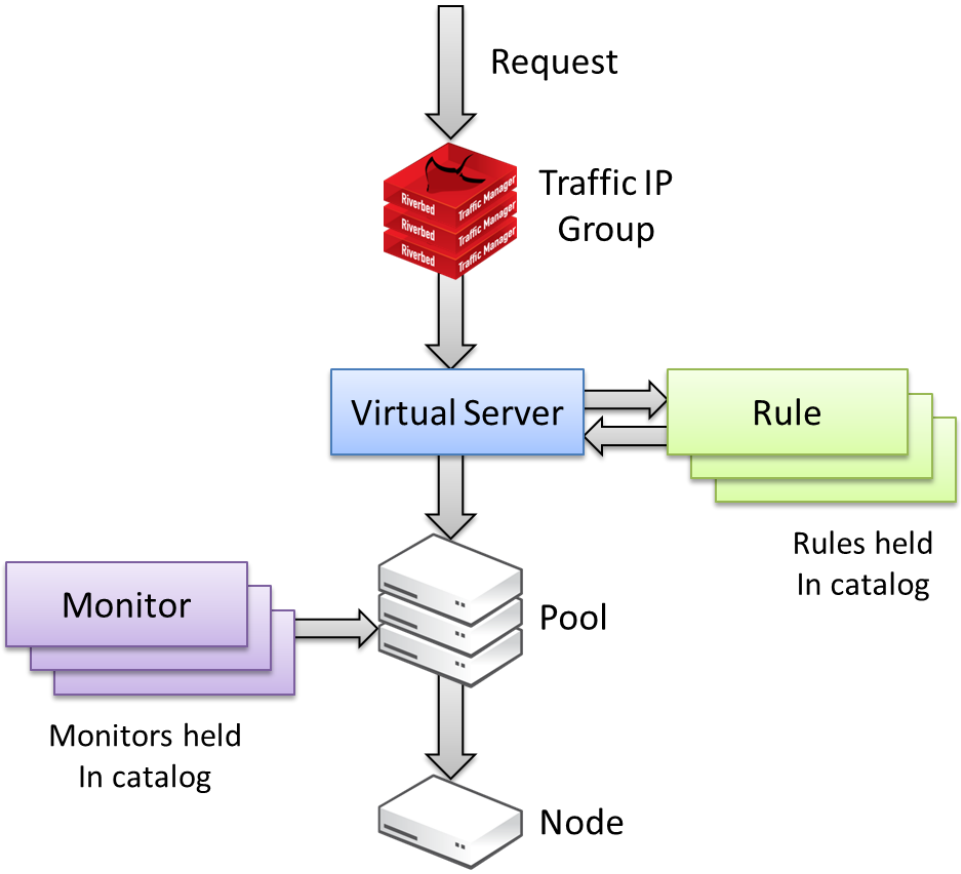


Fig. 16. Analyzing, modifying and monitoring traffic using TrafficScript rules

The TrafficScript rule can modify the request or response (for example, rewriting the URL or headers in an HTTP request), set session persistence parameters, or inform the traffic manager how to route the request to the most appropriate pool.

This makes it possible to control precisely how the traffic manager manages your traffic by using rules designed to meet your own hosting requirements.

TrafficScript Example

The following TrafficScript rule can be used with HTTP requests. It handles the request as follows:

Requests for `www.stingray.co.uk` are rewritten to `www.stingray.com`;

Requests for `.jsp` pages are routed to a set of application servers (the pool named `JSPServers`);

Requests for URLs beginning "`/secure`" are only allowed during office hours.

```
# Rewrite host header if necessary
```

```
if( http.getHostHeader() == "www.stingray.co.uk"
) {
  http.setHeader( "Host", "www.stingray.com" );
}

$path = http.getPath();

# Give .jsp requests to the "JSPServers" pool
if( string.endsWith( $path, ".jsp" ) ) {
  pool.use( "JSPServers" );
}

# Deny access to /secure outside office hours
if( string.startsWith( $path, "/secure" ) ) {
  if( sys.time.hour() < 9 || sys.time.hour() >=
18 ){
    connection.discard();
  }
}
```

The next rule can be used with HTTP responses. It processes the response as follows:

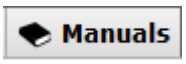
If the status code is 404 or 5xx, retry the request a maximum of 3 times;

If the response contains references to `www.stingray.co.uk`, rewrite it by changing these references to `www.stingray.com`.

```
$code = http.getResponseCode();

if( $code == 404 || $code >= 500 ) {

  if( request.getRetries() < 3 ) {

    # Avoid the current node when we retry, if
possible

    request.avoidNode( connection.getNode() );

    request.retry();

  } else {

    # Rather than returning an error, redirect
to /

    http.sendResponse( "302 Redirect",
"text/plain", "",

      "Location: /" );

  }

}



# We're only going to process text/html
responses, so

# break out of the rule if the response is of a

# different type...

if( http.getResponseHeader( "Content-Type" ) !=
          "text/html" ) break;



# getResponseBody automatically de-chunks and
uncompresses

# the response if required

$response = http.getResponseBody();



if( string.contains( $response,
"www.stingray.co.uk" ) ) {

  $response = string.regexsub( $response,
     "www.stingray.co.uk", "www.stingray.com",
"g" );

  http.setResponseBody( $response );

}
```

TrafficScript Documentation

The syntax and functions of the TrafficScript language are documented fully in the **TrafficScript Manual**, included with your distribution. It is also linked from the **Help** pages on the Admin Server, via the Manuals button.



The TrafficScript editing page also has a quick reference to the functions. When editing a TrafficScript rule, click **TrafficScript Help** to see this reference. The *Creating a Rule in the Catalog* section of CHAPTER 8 describes how to create rules.

Various applications of rules on a traffic manager

There are many occasions when you might use a rule:

Rules are used by a virtual server to choose a pool to handle a request. The rule can inspect any part of the request, possibly modify it, and decide which pool should handle the request.

You can use a rule to dictate session persistence information to a pool. After inspecting the request the rule can use the `connection.setPersistenceKey()` function to provide a string to persist on. This string is used by the Universal session persistence method to identify the session the request belongs to.

Rules can be used to check the response from the server and modify it, or even retry the request (if possible) if a transient error was detected.

Rules can override the classes assigned to a connection by the virtual server or the pool. This way, they can specify custom behavior for each connection; connections to a slow resource can be given a longer response time tolerance for example. Classes you can assign in this way include *Service Level Monitoring*, *Session Persistence* and *Bandwidth Management*.

Service protection classes can use rules. If you have associated a service protection class with your virtual server, it inspects the incoming packets. The class may use a rule to check the packet for a match with known web worms or viruses. This rule is executed before the main processing of the virtual server is carried out.

The session persistence and service protection applications of rules are covered in chapters CHAPTER 12 and CHAPTER 15 respectively.

Using a rule on a traffic manager

TrafficScript rules are stored in the Rules Catalog. You can create rules here, modify or duplicate them, and delete unused rules as required.

You can configure a virtual server to execute one or more rules from the catalog each time it receives a new request or response. This way, several different virtual servers can use the same rule, and modifications to the rule take effect on all virtual servers.

To use the TrafficScript functionality you need to:

1. Create a new rule in the catalog.
2. Configure your virtual server to use the rule.

Creating a Rule in the Catalog

1. Click the **Catalogs** button in the top menu bar. Go to the **Rules** tab.
2. Under **Create New Rule** enter a name for your rule in the text box. You have two choices for creating the rule: by using RuleBuilder, or by writing the rule directly in TrafficScript. Choose the method you require and click **Create Rule**.

Using RuleBuilder

Note: this section describes the full functionality of the RuleBuilder, however your license key may restrict the availability of certain Conditions, Actions, options, and the ability to use TrafficScript. If you encounter problems using some of the features described here, please consult your Support Provider who will be able to advise you on the best course of action to upgrade your software or license.

The RuleBuilder allows you to select conditions on the request to be examined, and actions that follow if any or all of these conditions are met.

You can add conditions based on the client's IP address and port, HTTP parameters, cookies, or the URL requested. Whenever you create a condition, extra text boxes appear for you to fill in the parameters for that item.

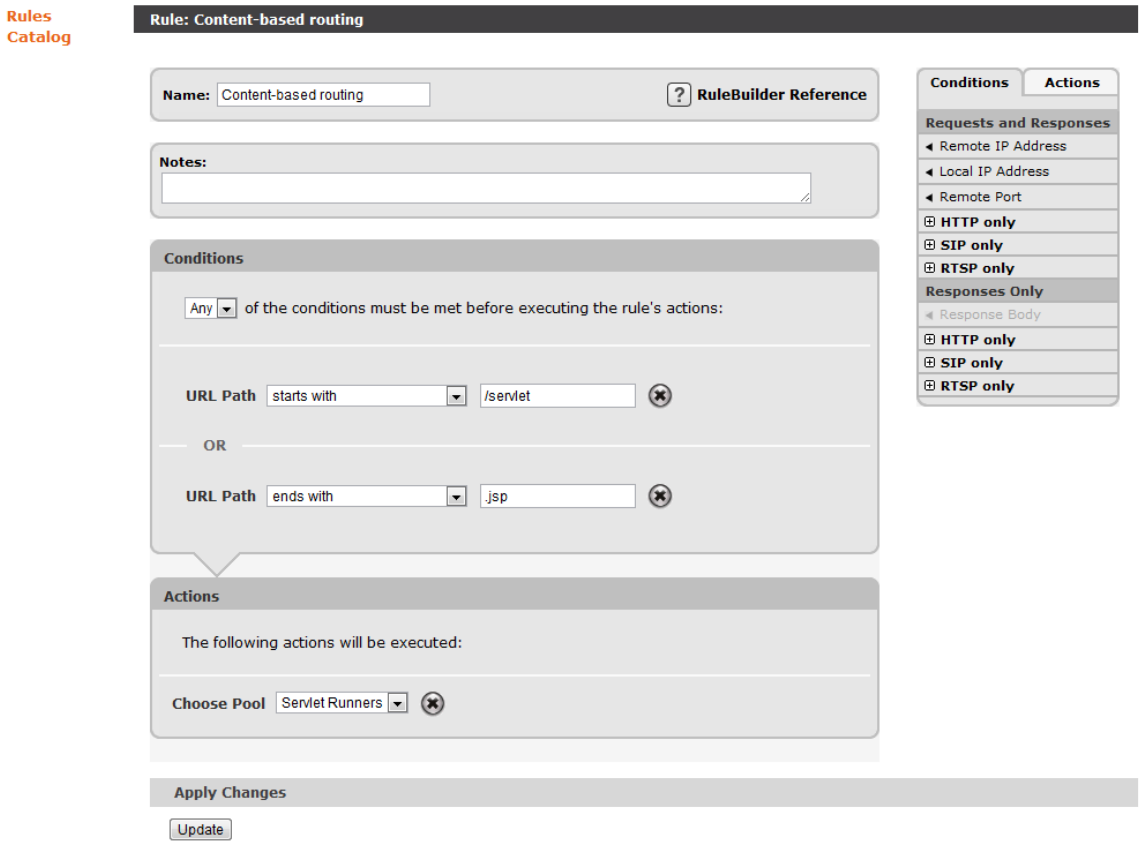


Fig. 17. Rulebuilder main screen

You can then select a series of actions for the rule to carry out. Parts of the request can be altered and the traffic manager can write log messages before a final action is performed. The possible final actions are to choose a specified pool, or to drop the connection; not more than one final action can be carried out, but a rule is not required to have a final action.

For example, you might construct a rule such as:

```
    IF
            URL Path  starts with
/servlet
    OR
            URL Path  ends with     .jsp
    THEN
            Choose Pool: Servlet Runners
```

Requests whose URL path does not start with `/servlet` or end in `.jsp` (such as requests for static HTML content) are ignored by this rule. A virtual server applying this rule can use a separate rule to deal with requests for other pages, or assign them to its default pool.

When you are happy with your rule, click **Update**.

You can also convert your rule to TrafficScript. The TrafficScript language gives you a much wider range and structure of possible conditions and actions. You can click **Preview Rule as TrafficScript** to see what the TrafficScript rule would look like, and use the **Convert Rule** button to convert your rule permanently.

When you begin to use the traffic manager's rules, you could start by using the RuleBuilder to implement simple rules. Examine the corresponding TrafficScript rule to familiarize yourself with the TrafficScript syntax. When you need to write a complex rule, you could use the RuleBuilder to prototype a simple version; once you have reached the limits of the capabilities of the RuleBuilder, proceed by converting the rule to TrafficScript for further editing.

Special variables

There are several special variables that can be included in the text boxes for RuleBuilder which are expanded to TrafficScript functions. These allow the generation of more dynamic rules, with extra flexibility.

For example, if you create a rule that sets the HTTP header 'X-Forwarded-For' to the value '%REMOTE_IP%', then the header will be set to the IP address of the client connecting to the traffic manager.

The special variables are as follows:

- %REMOTE_IP%– replaced with the remote IP of the client connection (TrafficScript function request.getRemoteIP());
- %REMOTE_PORT%– replaced with the remote IP of the client connection (TrafficScript function request.getRemotePort()).

Writing a TrafficScript Rule

The TrafficScript editing page shows a form where you can enter a text description of the rule, type the rule and check the syntax before compiling it. You may wish to write the rule in a separate text editor before pasting it in.

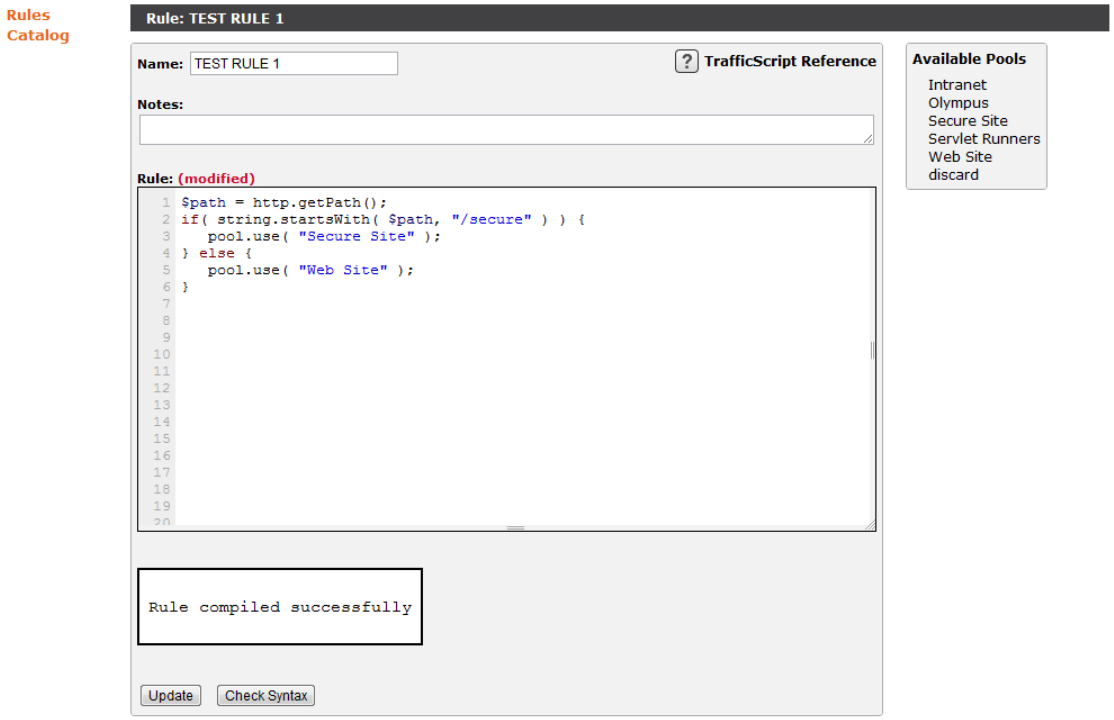


Fig. 18. Editing a rule in the TrafficScript editor

You can click the **TrafficScript Help** link to view the quick function reference, and use the **Check Syntax** button to check your rule. When you have finished click the **Update** button.

Note: Some TrafficScript functions and RuleBuilder conditions and actions are only appropriate in request rules or in response rules. For example, a function that modifies a parameter of a request will have no effect if used in a response rule (as the request has already been submitted to a node). Please refer to the online help or printed manual for descriptions of each TrafficScript function and RuleBuilder conditions and actions.

Applying a Rule to a Virtual Server

The new rule has been successfully created, but is not yet being used by any virtual servers. To configure a virtual server to use the rule, follow the instructions below.

1. Go to the **Virtual Servers > Edit** page for your virtual server. You can do this by clicking the **Services** button, then the **Virtual Servers** tab. Click on the name of your virtual server.

2. Now click on **Rules**. This presents you with a list of request rules and a list of response rules currently in use for that virtual server. You can choose new rules to add to each list from the drop-down selection at the end of each list.

3. Rules are executed in a specified order. If the first rule does not make a final decision about a request or response, the second rule is tested, etc. You can use the up and down arrows to move a rule within the list.

   For most protocols, you can specify whether the rule should be executed just once (against the first request or response), or against every request and response in the protocol dialogue.

   In RTSP and SIP the rules are automatically executed on each individual request or response that passes through the traffic manager.

   This option is not necessary for HTTP virtual servers because HTTP is a single request-response protocol, and requests within a keepalive connection are processed independently.

You can test the effect of a new rule by enabling and disabling it for your test virtual server.

Example Rules

Routing by Content Type

This example inspects the URL path in an HTTP request. It chooses a pool to handle the request based on the value of the URL path.

Set up a pool for each of these groups called `windows`, `sun` and `linux` respectively. The following rule directs network traffic according to the type of content.

```
$path = http.getPath();
if( string.endsWith( $path, ".jsp" )) {
  pool.use( "sun" );
} else if( string.endsWith( $path, ".asp"
)) {
  pool.use( "windows" );
} else {
  pool.use( "linux" );
}
```

Restricting Access Based on Time of Day

This example only allows access to a particular service during office hours (between 9am and 6pm, Monday to Friday). It discards all connections that occur outside these times.

```
$dayofweek = sys.time.weekDay();
$hourofday = sys.time.hour();


# $dayofweek: Sunday is 1, Saturday is 7
# $hourofday: office hours are between 9am
and 5:59pm
if( $dayofweek == 1 ||
    $dayofweek == 7 ||
    $hourofday < 9  ||
    $hourofday >= 18 ) {
  connection.discard();
}
```

In practice, it may be more appropriate to direct restricted traffic to a separate "error pool" of servers rather than just dropping the connection without warning. The servers in the error pool would be configured to return an appropriate error message before closing the connection. The procedure for doing this depends on the protocol being balanced.

Customer Prioritization

This example inspects the cookie in an HTTP request. It uses the value of the cookie to determine which pool to select. One pool is faster than the other because it contains machines reserved for premium users.

A company has a customer base divided into "gold" and "silver" membership. It wishes to give priority to the "gold" customers and has five servers, `yellow`, `green`, `blue`, `black` and `purple`.

Two pools are created: `standard`, for the "silver" customers, containing machines `yellow`, `green` and `blue`; and `premium`, for the "gold" customers, which includes

all five of the servers. Thus `black` and `purple` are only available to the "gold" customers.

The site uses a cookie login system, with the customer type encoded in the cookie. The traffic manager can differentiate between membership levels and send traffic to the correct pool:

```
$cookie = http.getHeader( "cookie" );
if( string.contains( $cookie, "gold" )) {
  pool.use( "premium" );
} else {
  pool.use( "standard" );
}
```

Managing Levels of Service

This example tags premium customers with a 'premium' service level monitoring class, and directs them to the 'premium' pool.

Non-premium customers can share the premium pool if the premium SLM class is functioning within its tolerance, but are directed to the 'standard' pool if the premium SLM class is running too slowly.

```
# $isPremium could be based on the presence
of a login

# cookie, the contents of the shopping
cart, or even the

# remote address the client is access from.


if( $isPremium ) {

  connection.setServiceLevelClass(
"premium" );

  pool.use( "premium" );

} else {

  if( slm.conforming( "premium" ) == 100 )
{

    pool.use( "premium" );

  } else {

    pool.use( "standard" );

  }

}
```

Routing Based on XML Traffic

TrafficScript includes support for parsing XML documents using XPath, an industry-standard language used to query XML documents.

XML documents are used by SOAP-based protocols such as Web services, and enable complex data to be exchanged and understood automatically without user intervention.

An XML document is organized into a tree structure of *nodes*[6]. Each node may contain a piece of data, or other nodes. XPath can navigate through these nodes to extract specific data from the XML document; this data can then be used by the traffic manager to make routing decisions on the traffic.

The XPath 1.0 specification is available at http://www.w3.org/TR/xpath.

Example: Google Search Request

The Google™ search engine has a Web services interface that accepts SOAP requests for search queries. A request for a search for Riverbed Technology would consist of an HTTP POST containing the following XML body data:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
  SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/
encoding/"
  xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-
instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
   <SOAP-ENV:Body>
      <namesp1:doGoogleSearch
xmlns:namesp1="urn:GoogleSearch">
         <key xsi:type="xsd:string">googleUniqueID</key>
         <q xsi:type="xsd:string">Riverbed Technology</q>
         <start xsi:type="xsd:int">0</start>
         <maxResults
xsi:type="xsd:int">10</maxResults>
         <filter
xsi:type="xsd:boolean">false</filter>
         <restrict xsi:type="xsd:string"/>
         <safeSearch
xsi:type="xsd:boolean">false</safeSearch>
         <lr xsi:type="xsd:string"/>
         <ie xsi:type="xsd:string">latin1</ie>
         <oe xsi:type="xsd:string">latin1</oe>
      </namesp1:doGoogleSearch>
```

[6] Note that these are unrelated to the traffic manager's back-end nodes.

```
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note that the SOAP body contains a "doGoogleSearch" node. This contains the parameters of the search request.

An Internet service may implement or proxy doGoogleSearch requests, and the traffic manager may be used to manage the traffic to this service.

For example, it may be necessary to split doGoogleSearch requests according to the specified maximum number of results. If maxResults is greater than 100, the request is to be sent to pool googleLarge; otherwise it should be sent to pool google.

A TrafficScript rule can use the functions xml.XPath.MatchNodeSet() and xml.XPath.MatchNodeCount() to query the SOAP request body and test XML nodes:

```
# Read the entire body of the SOAP/HTTP request
$body = http.getBody();

# XML parameters lie in the "urn:GoogleSearch" XML
# namespace:
$googlens = "xmlns:googlens=\"urn:GoogleSearch\"";

# Test for the presence of a "doGoogleSearch" node.
# If present, get the value of the "maxResults"
# parameter and choose the pool

if( xml.XPath.MatchNodeCount( $body, $googlens,
    "//googlens:doGoogleSearch" ) ) {

  $maxResults = xml.XPath.MatchNodeSet(
    $body, $googlens,
    "//googlens:doGoogleSearch/maxResults/text()"
);
```

```
  if( $maxResults >= 100 ) {
    pool.use( "googleLarge" )
  } else {
    pool.use( "google" );
  }
}
```

CHAPTER 9 TrafficScript Authentication Support

Note: Some versions of the product do not support TrafficScript, so this functionality may not be available. Full support can be obtained via a software or license key upgrade.

Overview

Stingray provides support for remote user authentication against external services. Each time a user connects to a service provided through the traffic manager, their credentials can be validated against the records held on a remote server.

To achieve this, you must first create remote authentication service definitions in the Authenticators Catalog (**Catalogs > Authenticators**). Authenticators created in this manner can be accessed through the `auth.query()` function from within a TrafficScript rule. This rule can then be added to a virtual server handling the service to be authenticated.

Configuring Authenticators

Stingray currently provides support for LDAP authentication only.

Remote LDAP authenticator records can be created on the **Catalogs > Authenticators** page of the Admin UI. To create a new authenticator, provide a name, host and port in the boxes provided. Clicking **Create Authenticator** will create the record, and allow you to access the remaining LDAP configuration keys. These are shown below:

Basic Settings

`Name`	The identifying name given to this authenticator. This name will be used in the `auth.query()` function call, within an authenticating TrafficScript rule. If the authenticator is renamed here, any rules

	referencing this authenticator will be automatically updated to use the new name.
Host	The host-name or IP address of the LDAP server to connect to.
Port	The port of the LDAP server to connect to.
Note	A description of this authenticator.

LDAP Settings

ldap!bind!dn	The Distinguished Name (DN) of the 'bind' user. The bind user is used to contact the LDAP server and search for the record belonging to the user being authenticated. The bind user must have permission to search for and read user records on the LDAP server. If no bind user is specified then the traffic manager will attempt an anonymous login to search for the user being authenticated.
ldap!bind!password	The password for the bind user specified by the ldap!bind!dn setting.
ldap!filter	A filter used to extract the unique user record located under the base DN. The string %u will be replaced by the username supplied when the authenticator is invoked. Examples of common LDAP filters include sAMAccountName=%u for Active Directory, or uid=%u for some Unix LDAP schemas.

ldap!filter!basedn	The base Distinguished Name (DN) under which the traffic manager will search for the record of the user being authenticated. The entries for all users that are to be authenticated by this LDAP authenticator must appear under the DN specified here. A typical base DN might be `OU=users, DC=mycompany, DC=local`.
ldap!attr	If the traffic manager finds a record for the user being authenticated on the LDAP server it can fetch back additional information from that record. This information can be used to perform additional checks on the user being authenticated, such as restricting access based on which group the user belongs to. To fetch back specific attributes from the user's record, a space- or comma-separated list of attribute names can be specified in the `ldap!attr` setting. To fetch back all attributes from the user's record, set `ldap!attr` to '*'. If the setting is blank, no additional attributes will be retrieved from the server. Any attributes retrieved from the user's record will be available in the return value from the TrafficScript function that requested the authentication.
ldap!ssl	This setting determines whether or not the connection to the LDAP server will be SSL-encrypted. The method by which the SSL

	connection is established can be specified in the `ldap!ssl!type` setting.
`ldap!ssl!type`	This setting specifies the method by which an SSL connection to the LDAP server is established. It is used only if `ldap!ssl` is set to **Yes**. The available methods are: • **LDAPS**: The traffic manager will establish a secure connection to the LDAP server before any LDAP messages are sent. • **Start TLS**: The traffic manager will use the LDAPv3 Transport Layer Security extension to establish a secure connection to the server.
`ldap!ssl!cert`	When connecting to the remote LDAP server over SSL, the traffic manager will ensure that the server's certificate is signed by the certificate authority specified by this setting. If the server sends a certificate that is not signed by the certificate authority specified here then an error will be returned to the TrafficScript function using the authenticator. If no certificate authority is specified then the server's certificate will not be validated.

Configuring the TrafficScript Rule

To use remote authentication within a virtual server, you should assign an authentication rule to it as a request rule. This rule should contain a call to the function `auth.query()` with the arguments shown here:

```
auth.query( authenticator, user, [password] );
```

This queries the named `authenticator` for information about `user` and, if supplied, checks that `password` matches the password on record for that user. It returns a hash containing two values, `OK` and `Error`, which are set according to the results of the verification. The result can also contain additional information returned by the authenticator, such as the Distinguished Name of the user that was queried. Please refer to the TrafficScript Reference Guide for more details.

TrafficScript rules are created on the **Catalogs > Rules** page of the Admin UI. For more details, please refer to CHAPTER 8.

The example below shows how you might use the `auth.query()` function to provide user verification based on a previously created Authenticator called '`ldap`':

```
# Verify the user's password using an LDAP
# authenticator called 'ldap'
$auth = auth.query( "ldap", $user, $pass );
if( $auth['Error'] ) {
   log.error(
      "Error with authenticator 'ldap': " .
      $auth['Error']
   );
   connection.discard();
} else if( !$auth['OK'] ) {
   # Unauthorised
   http.sendResponse( "403 Permission Denied",
      "text/html", "Incorrect username or
password",
      ""
   );
}

# Allow through members of the 'admin' group using
# the 'group' attribute returned by the
authenticator
if( $auth['group'] != "admin" ) {
   http.sendResponse( "403 Permission Denied",
      "text/html",
      "You do not have permission to view this
page",
      ""
   );
```

```
}
```

Configuring the Virtual Server

Once you have a rule with the appropriate settings configured, you must assign it to the virtual server on which you want to enable authentication:

1. Go to the **Services > Virtual Servers** section of the Admin UI and select the virtual server on which you want to enable authentication;

2. Click on the **Rules** section;

3. Under **Request Rules**, select your authentication rule and click '*Add Rule*';

CHAPTER 10 Java Extensions

This chapter gives an overview of the Java support available. Java can be called from TrafficScript, which allows for more powerful routing and easier traffic management.

For a more in-depth description, refer to the Stingray Java Development Guide.

Introduction to Java

Java is an object-oriented and platform-independent development language that has a large community of developers, libraries and applications. Stingray traffic management solutions allow the use of Java extensions in TrafficScript, offering greater flexibility in traffic manipulation. These Java extensions can then be invoked from TrafficScript rules.

The Stingray Java API is a way for customers and Internet Service Providers to extend the capabilities of the traffic manager.

Some examples of functionality available using the Java API in TrafficScript are:

- **Content processing:** Improved XML/HTML processing using specialized Java libraries.
- **Additional libraries:** ISV libraries supplied as a value-add solution, operating as 'pluggable' extensions to the traffic manager to add new features.
- **Authentication:** using RADIUS/TACACS+/LDAP etc, without the need to use an external web server.

Invoking a Java Extension

Java extensions are invoked from TrafficScript or RuleBuilder rules.

Whenever a Java extension is imported into the traffic manager, a basic RuleBuilder rule is created with the same name as the Java Extension. The rule invokes the extension, allowing it to inspect, modify and route the traffic. You can add this basic rule to the list of request or response rules run by a virtual server.

In many cases, a particular Java extension need not be called for every single request or response – for example, it only processes requests or responses of a particular

type. The Java Extension can be called conditionally by a RuleBuilder or TrafficScript rule:

```
if( http.getPath() == "/serverstatus" ) {
   java.run( "ServerStatus" );
}
```

Configuring the traffic manager to use Java

This section introduces the process of creating, configuring and running Java Extensions on the traffic manager. The Java Development Guide provides a much more complete description of this process.

Requirements

In order to use Java Extensions with the traffic manager, you must install the Sun Java run-time environment (JRE) version 1.5 or later. Previous versions are not supported by the traffic manager.

Stingray virtual/cloud instances include the necessary Java software to run Java Extensions.

Compiling a Java extension

To compile Java Extensions for the traffic manager you will need the following resources:

- Java Development Kit (JDK), which contains the Java compiler. The compiler can be downloaded from http://java.sun.com.
- Java Servlet API library: This can be found in `$ZEUSHOME/zxtm/lib/servlet.jar` or downloaded from the link in the **Java Extensions catalog**.
- Stingray Java Extensions API library: This can be found in `$ZEUSHOME/zxtm/lib/zxtm-servlet.jar` or downloaded from the link in the **Java Extensions catalog**.

To compile a Java Extension, run the command:

```
$ javac -cp servlet.jar:zxtm-servlet.jar
MyServlet.java
```

This will create a class file called *MyServlet.class*. It is assumed that you have copied *servlet.jar* and *zxtm-servlet.jar* to the directory you are compiling from (if not, you have to specify their paths as well).

You can also package up a Java extension along with any other needed classes in a single jar file. The traffic manager will automatically search jar files for Java extensions to use.

Loading Java Extensions onto the traffic manager

Java Extension class and jar files need to be added to the traffic manager's **Java Extensions Catalog**.

To upload the Extension, go to the **Catalog > Java Extensions** page and specify your class or jar file in the Upload section. If your extension depends on other Java jar files that are not included in the Java distribution, you should upload those into the catalog too.

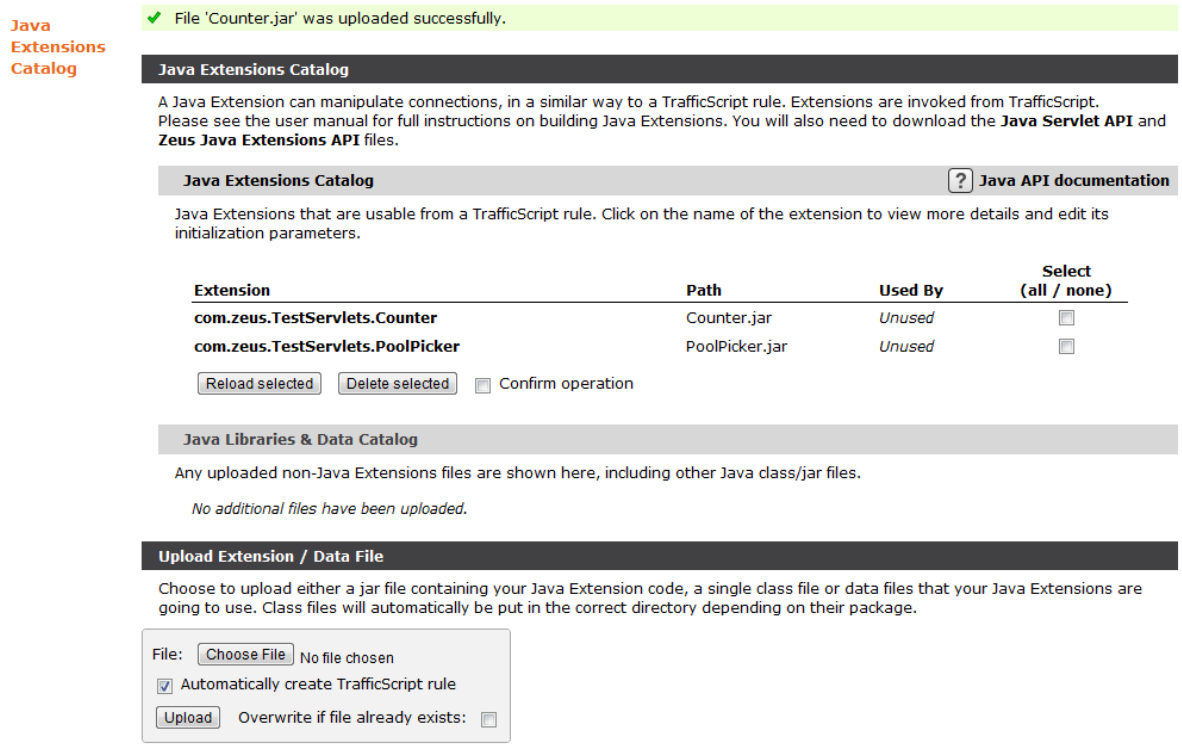


Fig. 19. Java Extensions Catalog page

Configuring the traffic manager's Java Extension runner

Stingray traffic management solutions include a Java helper application called the Java Extension Runner that hosts the Java extensions. You can control how the

traffic manager initializes and runs this helper application using the settings in the **System->Global Settings->Java Extension Runner** section of the Admin Interface.

To specify a Java executable, set the "java!command" field in the **[]** section of the interface to the name of the executable (and the path if it is not the default), along with any options the JRE should be run with.

- **java!command**: the path to the java executable, including any command-line arguments
- **java!classpath**: colon separated list of folders where the Java classes are located.
- **java!lib**: specifies a folder to search automatically for libraries used by your Java extensions. Note that the traffic manager will not load any Java extensions from this directory.
- **java!max_conns**: defines the maximum number of simultaneous Java requests allowed. Additional requests will be queued and will not be processed until the former ones have been completed. This setting is per CPU; for multicore processors you will have to multiply this setting by the actual number of processors.
- **java!session_age**: value in seconds, defining a timeout to maintain a java session.

To check the setup, press the **Diagnose** button, go to the **Cluster diagnosis** tab, and verify that the "Java extensions" line is green. The traffic manager should now be ready to run Java extensions.

CHAPTER 11 Protocol support

Stingray traffic management solutions load-balance and process application traffic that is enclosed in TCP connections or UDP datagrams. The majority of features, including TrafficScript, load balancing, session persistence, bandwidth management etc can be applied equally to all protocol types.

Where appropriate, additional TrafficScript functions specific to a particular protocol are included, making management and control of traffic in that protocol easier.

Basic TCP Protocols

'Generic Server First', 'Generic Client First' and 'Generic Streaming' protocols are the most basic L7 protocol types that the traffic manager can use. They are useful when managing custom protocols or simple TCP connections because they do not expect the traffic to conform to any specific format.

Server-first protocols

Server-first is the simplest TCP protocol type. When the traffic manager receives a connection from a client, it immediately runs any TrafficScript request rules, then immediately connects to a back-end server. It then listens on both connections (client-side and server-side) and passes data from one side to the other as it comes.

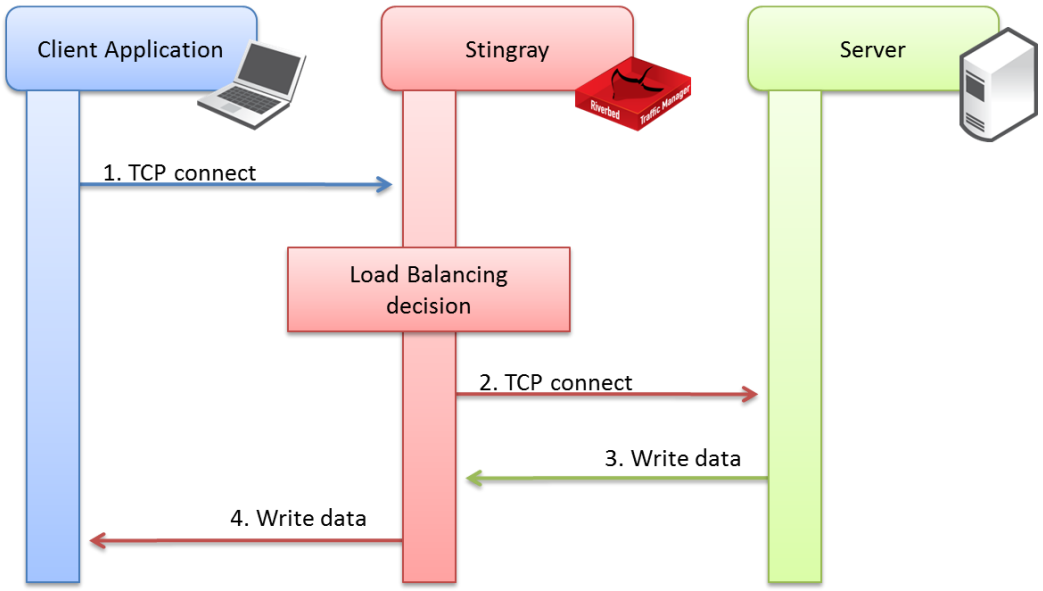


Fig. 20. Server-first protocol load balancing

Server-first protocols include one-shot protocols such as Time (where the server writes the response data, then immediately closes the connection), or complex

protocols like MySQL (where the server opens the dialog with a password challenge).

The virtual server protocol type 'Generic Server First' is most suitable for protocols where the server 'speaks' first. In practice, the design of most protocols means that 'Generic Client First' is more appropriate.

Client-first Protocols

Client-first is a modification of server-first, appropriate for protocols where the client connects and sends a request before the server replies. You can use the "Generic Client First" protocol type to inspect a client's request and select a server pool that should be used to select the server node to be used.

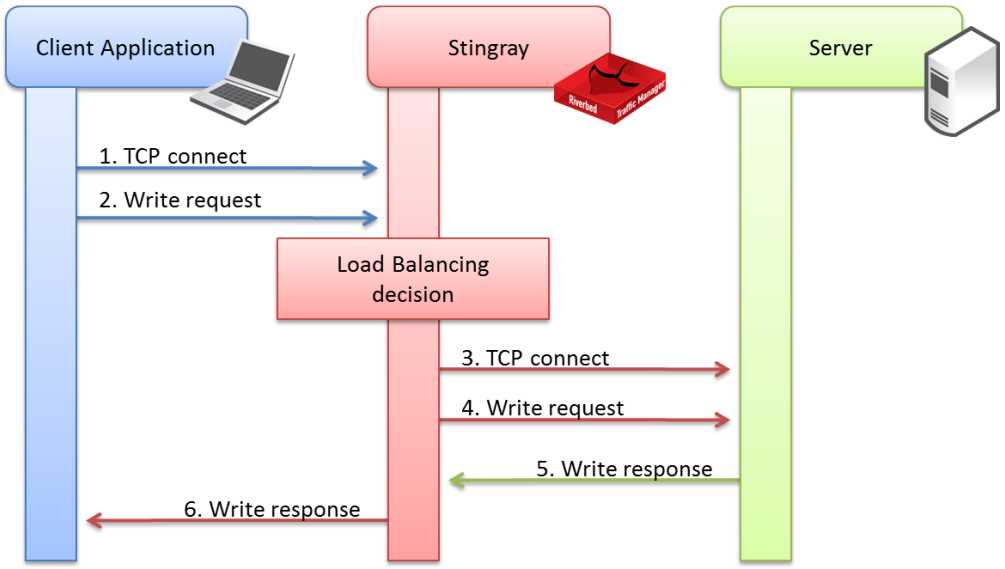


Fig. 21. Client-first protocol load balancing

The traffic manager is only alerted when a client connection has been established and data has been received. The traffic manager then proceeds in the style of server-first, i.e. runs TrafficScript rules, connects to back-end and relays data back and forth.

Rules processing in detail

The traffic manager alternates between running TrafficScript request and response rules. It will run the request rules when the first data from the client received; the request rules may block if they use functions like `request.getLine()` or `request.get()` to read further data from the client.

Once the request rules have completed, the traffic manager will forward any further data it receives from the client on to the server. The traffic manager will run the response rules as soon as it receives any data (i.e. a "response") from the server; these response rules may also block.

The traffic manager will then continue to forward any further data from the server to the client; the traffic manager will wait for any data from the client and run the Request rules again; in this manner, the traffic manager switches between waiting for request data to run request rules, and waiting for response data in order to run response rules.

Note that rules that are configured as “Run Once” rather than “Run Every” will only be run on the first server or client data, not on subsequent iterations.

Server-first with 'server banner'

Server-first with a 'server banner' is a different optimization for ‘server-first’ to cater for servers which broadcast a banner on connect, such as SMTP. ‘Server-first with server banner’ allows you to inspect and make a load-balancing decision based on the client’s request.

When a client connects, the traffic manager immediately writes the configured 'server-first banner' to the client, then proceeds as a regular client-first connection. In addition, the traffic manager slurps and discards the first line or data (terminated by a newline) that the server sends.

Once again, you can use TrafficScript rules to inspect the client's request data before making your load-balancing decision.

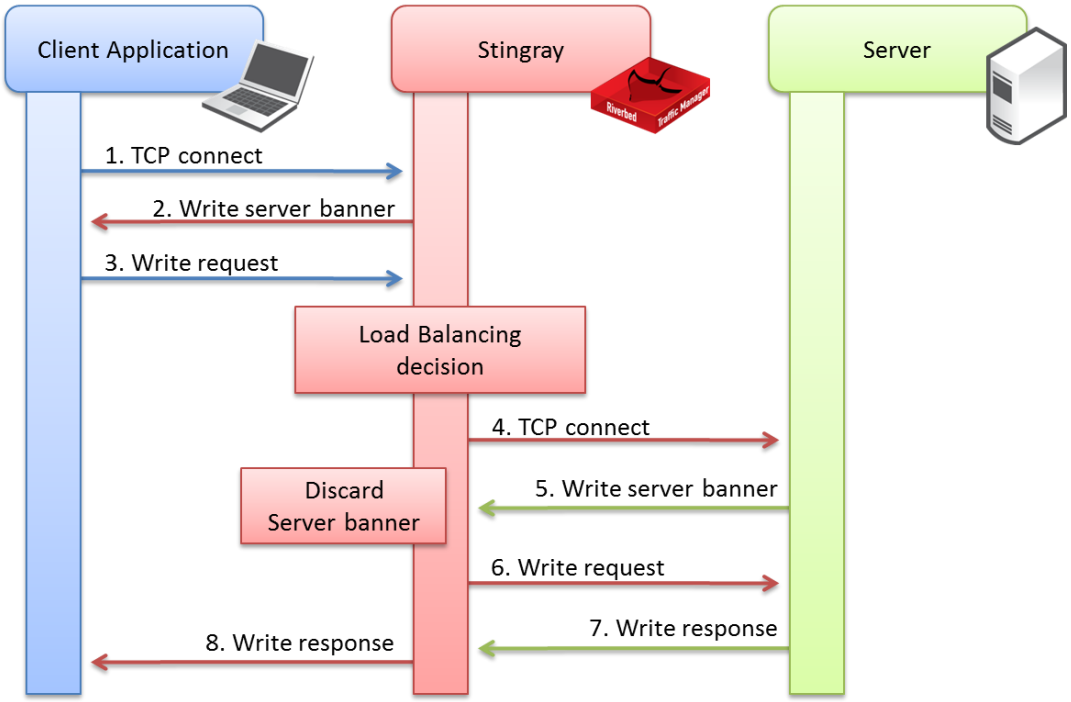


Fig. 22. Server-first load balancing with Server Banner

To configure this behavior, select the 'Generic Server First' protocol type and configure a banner message in the **Connection Management** settings in the Virtual Server configuration.

Generic streaming protocols

If you are using a protocol that does not require the server to respond to every message that the client sends, or if you need extra flexibility when processing data in TrafficScript, you can choose the 'Generic streaming' option. This server type allows either the client or server to send the first message when a connection is established and also allows TrafficScript rules to be invoked whenever data is received on the connection.

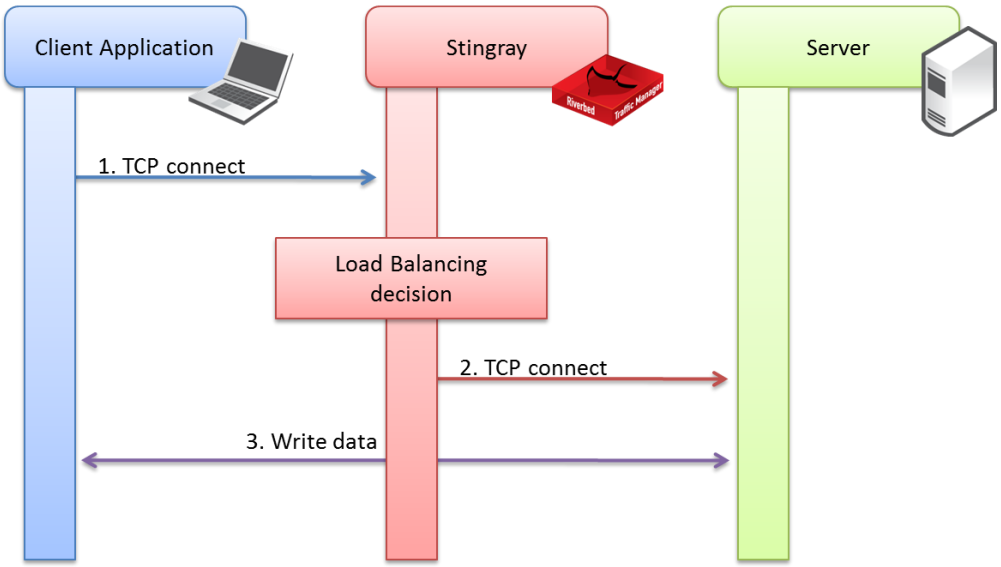


Fig. 23. Load balancing with generic streaming protocols

This option is for protocols where there is no request-response semantic. Either side of the connection can write the first message, with no response being necessarily required or expected. TrafficScript request rules are run whenever the client writes data on the connection, and similarly response rules are run every time the server writes data on the connection.

HTTP

The traffic manager's HTTP virtual server protocol type contains a number of optimizations and specializations for HTTP traffic.

- The traffic manager manages client-side and server-side connections independently, re-using keepalive connections on the server side whenever possible to reduce the number of established and new TCP

connections to the server. This minimizes the number of concurrent connections the servers need to handle, and brings big performance and capacity gains.

Note that features such as HTTP POSTs and missing content lengths can make keepalives unsafe to use; the traffic manager detects when this occurs and creates new connections appropriately. NTML authentication specifically requires that keepalives are enabled.

- The traffic manager conceals the use of keepalives and pipelining from the administrator, so that traffic management rules need only concentrate on the simple request-response nature of an HTTP transaction. Every HTTP transactions is processed and handled independently, regardless of whether or not it is in a keepalive connection.
- The traffic manager automatically handles HTTP encodings, such as gzip and deflate content compression and chunked transfer encoding. For example, if you inspect an HTTP response using the `http.getResponseBody()` TrafficScript function or from within a Java Extension, the traffic manager will automatically de-chunk and uncompress the response so that it can be easily manipulated.
- The traffic manager includes a large set of specialized HTTP TrafficScript functions that make it easy to process HTTP requests and responses. For example, functions are provided to manipulate HTTP cookies, read HTTP headers (such as the host header) and process URLs without having to understand the underlying encodings, variations in format and protocol details that complicate these tasks.

SSL

When you configure the traffic manager to manage traffic using the SSL protocol, the traffic manager does not automatically decrypt the traffic. The SSL protocol type is used for SSL pass-through, without modification.

You can use the 'SSL Session ID' persistence type to load-balance SSL connections across a pool, ensuring that connections with the same SSL session ID are sent to the same back-end server. This is an important consideration because SSL servers do not

generally share SSL session IDs with each other. So if a client attempts to re-use an SSL session ID, but is directed to a new server, it would then need to renegotiate its SSL session. This is compute-expensive and will add load to your SSL server farm. In fact, without SSL session ID persistence, adding more SSL servers to your cluster can reduce performance because SSL sessions need to be renegotiated more frequently.

Note: if you want to decrypt SSL traffic and process it using the internal protocol (such as HTTP), you should configure your virtual server to use the internal protocol, and enable SSL decryption in the virtual server configuration settings (see the *SSL Decryption* section of CHAPTER 4, and CHAPTER 13, for details on configuring SSL).

SSL connection renegotiation protection

Both SSLv3 and TLS (Transport Layer Security) protocols allow either the client or server to initiate renegotiation of the secure connection. This might be to establish new cryptographic parameters by which the connection will be governed.

During renegotiation, the secure connection becomes potentially vulnerable to interception by an attacker who might seek to inject traffic as a prefix to the client's interaction with the server. This mechanism, known as a Man-in-the-Middle (MitM) attack, can be used to set up a new client connection from the attacker, fooling the target server into believing that the initial data transmitted by the attacker is from the same entity as the original client.

Furthermore, renegotiation can be used to carry out a Denial of Service (DoS) attack because it poses a much higher load on the server than on the client. A client could simply request a re-handshake at a very high rate causing a very high computational burden on the server.

RFC-5746 (http://tools.ietf.org/html/rfc5746) introduces a TLS extension that prevents the MitM attack by cryptographically tying renegotiations to the client that is performing it. However, the introduction of this extension does not necessarily prevent or protect from a DoS attack.

Stingray provides a configuration setting (`ssl!ssl3_allow_rehandshake`) that allows you to select how a potential renegotiation handshake should be handled. You can find it on the **System > Global Settings** page of the Admin UI, within the **SSL Configuration** section. The available options are:

Always allow	Renegotiation is always allowed, even if the client lacks support for RFC-5746, but without any form of protection against intervention by a third

	party. This setting is NOT RECOMMENDED, and could leave your system open to MitM or DoS attacks.
Allow safe re-handshakes [default setting]	This includes the case in which the client supports RFC-5746 and the case that no data has yet been transmitted over the connection. Since the aforementioned attack consists of pre-pending plain text, if no data has yet been received, the re-handshake is safe. This is of limited use as from the client's perspective, the re-handshake is actually the first handshake, so the client cannot know whether the connection is under attack. This option is therefore mostly provided for backward compatibility with clients that have yet to be upgraded to support RFC-5746. Note that since this setting allows re-handshakes it still leaves an installation potentially open to the DoS attack mentioned above.
Only if client uses RFC 5746 (Secure Renegotiation Extension)	Renegotiation handshakes are only allowed if the client supports *Secure Renegotiation* as defined in RFC-5746. Note that since this setting allows re-handshakes it still leaves an installation potentially open to the DoS attack mentioned above.
Never allow	Reject all attempts at a renegotiation handshake. This is the most secure method as it protects against both MitM and DoS attacks. However, service failure is possible if, for example, some aspect of your installation depends on client-side renegotiation.

SMTP (Simple Mail Transport Protocol)

SMTP is a server-first protocol, but with one additional capability. For improved security, an SSL client can request that an SMTP connection is encrypted using SSL (also known as TLS).

If you configure a virtual server to use the SMTP protocol, it will behave just like a regular 'Generic Server First' protocol (see the *Server-first protocols* section of CHAPTER 11), and you will probably want to configure a server first banner (see the *Server-first with 'server banner'* section of CHAPTER 11).

However, if you enable SSL for an SMTP virtual server, you will find an additional setting named **ssl!expect_starttls** in the **Virtual Server > SSL** configuration page:

- If **expect_starttls** is set to 'no', the virtual server will decrypt SSL traffic in the normal way, i.e., expect traffic to be SSL-encrypted at the very beginning of the connections;
- If **expect_starttls** is set to 'yes', the traffic manager will process the traffic in plain text, as if it were unencrypted. The traffic manager will watch for a 'STARTTLS' SMTP command from the client and will then initiate an SSL handshake to upgrade the client's connection to SSL.

The traffic manager will forward all traffic on to the back-end SMTP servers unencrypted (in plaintext) unless you enable SSL encryption in the Pool configuration. If you do this, the traffic manager will encrypt the connection from the outset, rather than using the STARTTLS SMTP command.

FTP

The FTP protocol is used to transfer files from client to server. An FTP session consists of:

- a control connection, initiated by the client to port 21 (typically) on the remote server; this connection is received and proxied by the traffic manager
- one or more data connections that are created on-the-fly as a result of commands sent down the control connection; these connections may be initiated by the server to the client ("active mode") or by the client to the server ("passive mode").

The traffic manager's 'FTP' virtual server type includes a full FTP proxy that intercepts and manages the requests to create additional data connections. The traffic manager proxies the data connections just as it proxies the control connection, supporting both the active and passive modes of data transfer.

The proxy fully supports all commonly used FTP data connection commands, including the long (RFC 1639) and extended (RFC 2428) versions. The proxy does not support the RFC 4217 encryption control statements ('AUTH TLS', 'AUTH SSL', 'CCC'), and removes these commands from the control stream.

Note: **consider source port** should not be used with any Traffic IP Addresses that are used by FTP virtual servers (see the *Creating a Traffic IP Group* section of CHAPTER 6).

FTP Security

FTP is not a secure protocol. Passwords are not used for 'anonymous' file transfers which are common across the public internet, but where passwords are needed, they are sent in plain text and vulnerable to eavesdropping. If this is of concern, alternative protocols such as SCP, WebDAV or SSL-wrapped FTP (see the *SSL-wrapped FTP* section of this chapter) must be used.

A second concern is the ease with which an eavesdropper can snoop on the dialog that prepares the client and server for a data connection, and can then step in and initiate the connection to the listening party.

For example, when an Active FTP connection is set up, the client informs the server of the local client port that the server should connect to in order to establish the data connection. An eavesdropper could step in and connect to the client's local port before the server does, and then capture any data the client sends (or return a fake file to the client).

The FTP protocol does not have any built-in security measures to authenticate connections and to prevent this from happening, but many modern FTP clients and servers will check the source IP address of incoming connections to verify that they originate from the remote FTP agent they are communicating with. This behavior is enabled by default in the traffic manager – go to **Virtual Server > Connection Management** and examine the settings in the **FTP-Specific Settings** section of that page:

- **ftp_force_client_secure** (default 'yes'): verify that all passive data connections originate from the client's IP address;
- **ftp_force_server_secure** (default 'yes'): verify that all active data connections originate from the server's IP address;

- **port_range**: specify the range of destination ports used for data connections; these ports may need to be permitted through intervening firewalls so that, for example, clients can make passive connections to the traffic manager.

FTP Source Ports

The FTP specification recommends that FTP server data connections should use a source port one below the FTP service port. The typical FTP service port is port 21, so the data source port is recommended to be port 20.

In practice, the vast majority of FTP clients and servers ignore this recommendation because it has security implications and does not provide any additional assurance or authentication. By default, the traffic manager will select high port numbers on a random basis for FTP data connections that originate from the traffic manager.

Specifying the Source Port

You can use the **ftp_data_source_port** virtual server setting (in the **Connection Management > FTP Settings** section) to specify a source port that must be used. This is often required when an upstream firewall is used to block outgoing connections, other than those permitted with an explicit policy (for example, source port and IP).

FTP software in a Unix/Linux environment requires certain permissions to issue requests from low ports (lower then 1024), such as port 20. By default, the traffic manager drops these permissions early (principle of least privilege), so attempts to configure **ftp_data_source_port** to a low value will fail. If you wish to use low ports, enable the setting **ftp_bind_data_low** in the **System Settings** section of the **Global Settings** page; this will cause the traffic manager relinquish fewer permissions so that it retains the ability to bind to low ports.

On modern Linux platforms only the privilege to bind to low ports is kept, while on Unix variants full root privileges will be retained.

SSL- wrapped FTP (FTPS)

Stingray traffic management solutions support the pre-RFC 4217 use of FTPS (referred to as 'Implicit FTPS' or 'SSL-wrapped FTP'). In this implementation, the control port is encrypted from the very beginning; the data connections are generally encrypted but can operate in plaintext if desired.

The virtual server protocol type for SSL-wrapped FTP is 'FTP'; the virtual server and pool can be independently configured to decrypt or encrypt the control channel communications.

Controlling the control channel

- **ssl_decrypt**: Enable the **ssl_decrypt** setting on the **Virtual Server > SSL Decryption** page to enable SSL decryption for client-side connections. Refer to the *Setting up SSL Decryption* section of CHAPTER 13 for more information on configuring SSL decryption.

- **ssl_encrypt**: Enable the **ssl_encrypt** setting on the **Pool > SSL Settings** page (see the *SSL Encryption* section of CHAPTER 13) to encrypt the control connection before sending it to a back-end server.

The traffic manager's FTP proxy disables the encryption control statements ('AUTH TLS', 'AUTH SSL', 'CCC') in RFC 4217 so that they cannot interfere with the established SSL-wrapped FTP session.

Controlling the data channel

The traffic manager can decrypt the data channel or leave it unmodified: this behavior is controlled by the **Virtual Server > SSL Decryption** setting **ftp!ssl_data**.

- If enabled, **ftp!ssl_data** will cause the Virtual Server to decrypt all data connections using the same public certificate as is used for the control port. Note that the FTP Virtual Server must also be configured to decrypt SSL with **ssl_decrypt**.

- If disabled, the Virtual Server will pass all data through unmodified. If the back-end server wishes to use SSL on the data connection, the client can negotiate directly with the back-end server, with the traffic manager as an intermediate proxy.

Note: on the server side, the traffic manager will encrypt the data connection if either:

a) The Virtual Server setting **ssl_decrypt** is disabled (the Virtual Server is not decrypting SSL), and the Pool setting **ssl_encrypt** is enabled (the Pool is encrypting the server-side of the data channel). In this case, data communications with the clients will be in plaintext.

b) The Virtual Server settings **ssl_decrypt** and **ftp!ssl_data** are both enabled (the Virtual Server is decrypting SSL), and the Pool setting **ssl_encrypt** is enabled (the Pool is encrypting the server-side of the data channel). In this case, data communications with the clients will be encrypted.

Use cases for SSL-wrapped FTP

Use case 1: servers and clients support SSL-wrapped FTP

In this use case, all parties support and use SSL-wrapped FTP and the traffic manager acts as a load-balancing proxy, connecting clients to servers. The traffic manager decrypts and re-encrypts the control connection internally so that it can be inspected, and the traffic manager can optionally decrypt and re-encrypt the data connection.

- Enable SSL decryption (Virtual Server: **ssl_decrypt**) and SSL encryption (Pool: **ssl_encrypt**) so that the traffic manager can inspect and proxy the control channel, and create data connections on the fly.
- Optional: enable **ftp!ssl_data**. If disabled, clients and servers can negotiate whether or not to use encryption on the data connections. If enabled, the traffic manager will require clients to use SSL for data transfers.
-

Note: Enabling **ftp!ssl_data** brings two benefits:

1) Certificate management is easier as all connections are encrypted and authenticated against the traffic manager Virtual Server certificate;

2) The traffic manager will force all clients to use SSL for their data connections.

Without **ftp!ssl_data**, you would need to synchronize certificates across the traffic manager and the back-end servers, and you could not prevent clients and servers from using plaintext (unencrypted) data connections. However, these benefits come at the cost of additional SSL processing on the traffic manager system.

Use case 2: clients support SSL-wrapped FTP; servers do not

This configuration can be used with FTP servers that do not support SSL or in situations where the traffic manager and FTP servers are in close proximity and there is no need to secure the network connections between them:

- Enable SSL decryption (Virtual Server: **ssl_decrypt**), but do not enable SSL encryption (Pool: **ssl_encrypt**). The traffic manager will decrypt SSL-wrapped FTP from the client, and pass it in plain-text to the FTP servers in the pool.

- Recommended: enable **ftp!ssl_data**. If disabled, clients and servers will always use plaintext (unencrypted) connections for data transfer. If enabled, the traffic manager will force clients to use SSL for data transfers.

This use case illustrates an easy way to upgrade your FTP services to SSL-wrapped FTP without requiring any server modifications.

Real Time Streaming Protocol

The Real-Time Streaming Protocol (RTSP) allows a client to establish and control either a single or several time-synchronized streams of continuous media, such as audio and video servers. Applications based on RTSP act as a video-like remote control panel for multimedia servers. Some servers also use the Session Initiation Protocol (SIP) combined with RTSP in the same conference.

RTSP is used by client applications such as Quicktime, MPlayer, VLC, Windows Media Player and RealPlayer.

The traffic manager is able to recognize and manage RTSP traffic, allowing users to load balance a pool of media servers that handle this type of traffic.

The figure below shows the usual layout of a RTSP connection. An RTSP client will establish a TCP connection to an RTSP server, via the traffic manager, over which RTSP requests will be issued. This is the control channel. The media data itself may be sent independently over UDP (the most common scenario), or multiplexed over TCP using the same connection.

Alternatively HTTP may also be used for data delivery. Note that the client and server negotiate during the setup if the underlying transport protocol to be used is UDP or TCP; this configuration is not done from the traffic manager.

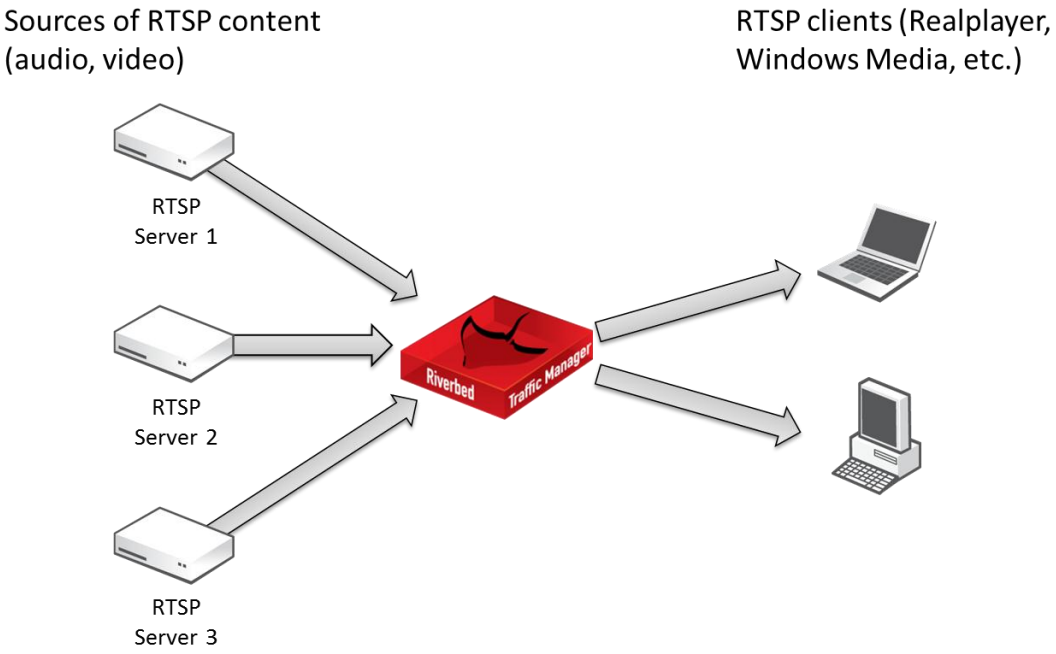


Fig. 24. A traffic manager acting as a load balancer for standard RTSP content traffic

RTSP itself only *controls* the transport of the media. The protocols involved in a full media transaction may include the following:

- **RTP:** Real-time Transport Protocol, used to transport the data.
- **RTCP:** Real Time Control Protocol, used to provide extra information for an RTP stream.
- **RDT:** Real Data Transport, a proprietary data delivery format from Real Networks that transports the data stream.

From a functional point of view, the use of a traffic manager for the handling of RTSP:

- Reduces the stress on the servers and allows load balancing across a pool of RTSP servers;
- Customizes and improves the performance, by using TrafficScript rules;
- Provides session persistence in the case of a server failure.

Setting up an RTSP service

RTSP runs over port 554 by default. To manage an RTSP service with the traffic manager, create a Virtual Server listening on port 554 (or an alternative port if appropriate), and select the protocol type 'RTSP'.

Your virtual server will need to use additional ports to stream content over UDP. If you need to specify the range of ports which will be used, use the **Connection Management** settings for the RTSP virtual server. In the **RTSP-Specific Settings** section you may modify the port range. It is recommended that you enter the largest range that you can.

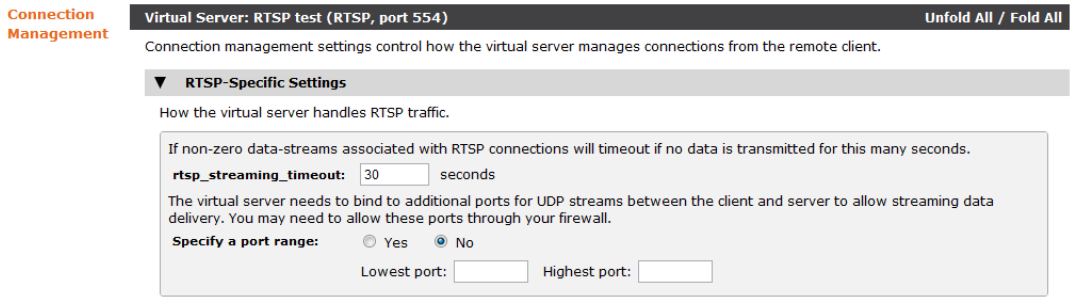


Fig. 25. Configuring the port range for RTSP

The total number of ports needed depends on the transport method and specific servers used, the specific player used by the clients and the format of the files to be streamed. For example, in a system with 100 active clients, up to 8 ports per client might be needed, giving us a total number of 800 ports.

Session Initiation Protocol

Session Initiation Protocol (SIP) is an application-layer control protocol that can initiate, control, modify or terminate sessions with one or more participants. SIP is typically used for telephone and video calls, streaming multimedia distribution and conferencing. The SIP protocol offers the following features:

- User location
- User availability
- User capabilities
- Session setup
- Session management

SIP features

This section outlines the features of SIP in more detail.

- **User Location:** When a user activates a SIP device (e.g. a SIP phone), it registers its current location with a SIP server. The user can then be contacted at the server's domain. Each user can have several devices in different locations registered with a server at the same time.

  When someone attempts to contact the user, the message will arrive at the server. The server will then choose one or several of the user's registered locations to forward the message to.

- **User Availability:** When a SIP device receives a message it can respond in several different ways. The response might be determined by user input – for example the user answering an incoming call. Alternatively, the response might be automatically generated, for example if the device is already in use a 'Busy' response might be returned. The server will decide what to do with this response - it might send it back to the caller or it might try sending the original message to a different location.

  If the user has no devices registered with the server then they are considered to be unavailable.

- **User Capabilities:** SIP devices are able to query other SIP devices to learn what capabilities they have. Communication can then take place using the features that both devices support.

- **Session Setup:** SIP can be used with the Session Description Protocol to establish a direct communication channel between two devices. This communication channel is then used to transmit session data, such as voice data in the case of a telephone call, while the SIP channel is used to send control information, such as a request to disconnect the call. Session data is typically transmitted using the Real-time Transport Protocol (RTP).

- **Session management:** SIP controls termination of sessions and transfer of sessions between different devices. It is also possible to change the parameters of an established session using SIP.

SIP supports both IPv4 and IPv6, and operates over TCP and UDP.

Stingray traffic management and the SIP protocol

The traffic manager load-balances SIP traffic between SIP proxy servers, creating a SIP infrastructure with high availability, reliability and extensibility. The figure below shows an example of a SIP network that includes a traffic manager. The SIP phones are typically located at the users' individual locations, while the traffic manager, the SIP proxy servers and the location database are owned by a SIP service provider.

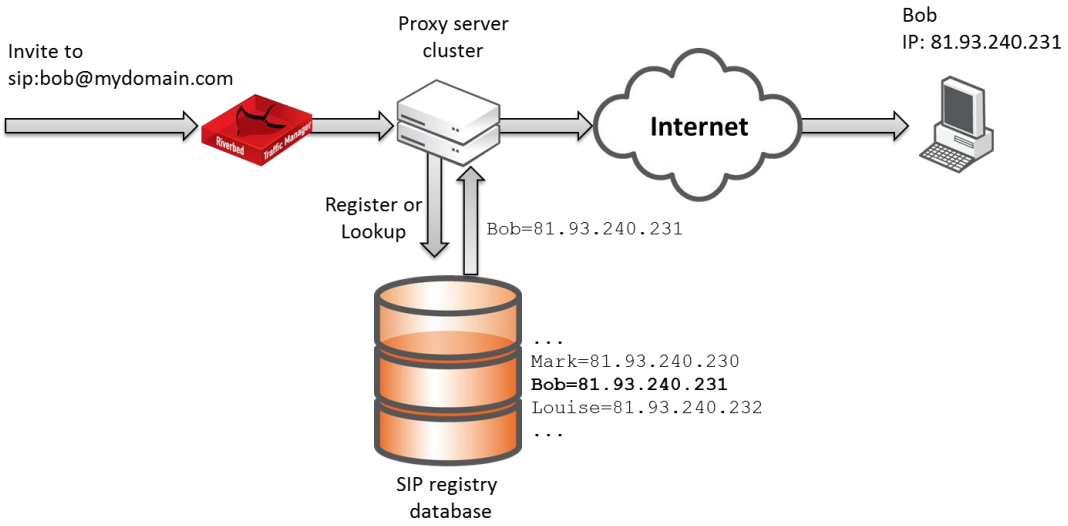


Fig. 26. Detail of how an IP phone call using SIP would work with a traffic manager

When working with SIP traffic, Stingray provides the following features:

- Load-balance SIP requests between SIP proxy servers with customizable load-balancing algorithms.
- Maintain session persistence so all requests that are part of the same session will be sent to the same proxy server.
- Modify SIP request and response packets as they pass through the traffic manager using RuleBuilder and TrafficScript rules.
- Issue responses to incoming requests without sending them to a proxy server.
- Monitor the health of the proxy servers in use and direct traffic away from them when they go down.
- Act as a gateway for all SIP data and session data when all clients are in a private network.

- Protect against Denial of Service attacks and malicious data.

Configuring the proxy servers to support traffic management

The traffic manager is most effective when used in front of a series of SIP proxy servers, such as Oracle/Weblogic SIP server or SIP Express Router (SER). To work in this configuration, the proxy servers must recognize the domain clients use to contact the traffic manager as one they are responsible for. For example, if sip:user@stingray.com resolves to the IP of the traffic manager, then the proxy servers must be responsible for the domain stingray.com. The proxy server will then process the request instead of forwarding it to that domain, which would result in the request looping between the traffic manager and the proxy server.

Alternatively, if your proxy servers do not support domains, go to the **Virtual Servers > Edit > Connection Management** page of the Admin UI and enable the "rewrite_request_uri" option. Enabling this option will tell the traffic manager to replace the URI in each request it receives with the URI of the proxy server it sends the request to.

Setting up a SIP service on the traffic manager

A new SIP service can be created through the **Manage a new service** wizard. Specify a suitable name for the service, and choose either the SIP over TCP protocol or the SIP over UDP protocol. The default SIP port is 5060 for both TCP and UDP services. This wizard will set up a new SIP virtual server and pool with the details provided.

For your SIP virtual server to operate correctly, it must listen to all the IP addresses used to send or receive a SIP request. To do this, navigate to the **Virtual Servers > Edit > Basic Settings** page and locate the **Listening on** setting. Ensure that "All IP addresses" is selected (this is the default).

You can also use both the UDP and the TCP protocols on the same port by creating two virtual servers and assigning one of the protocols to each virtual server.

Firewalls

Important: When using the traffic manager in a SIP infrastructure, SIP requests are sent to the client on a different connection to the one used when they register with a SIP proxy server. As a result, the client's firewall must be configured to allow SIP requests through to the phone. Some firewalls will do this automatically, whereas others will need to be configured to forward requests on port 5060 to the user's phone.

SIP operation modes on the traffic manager

A SIP virtual server can run in three different operational modes: **SIP Routing, SIP Gateway** and **Full Gateway.** This setting affects how much involvement the traffic manager has over control information (SIP messages) and session data (typically RTP data in voice and video communication).

To define the operational mode of the SIP server, edit your server in the Stingray admin interface, then click on **Connection Management>SIP-specific settings.**

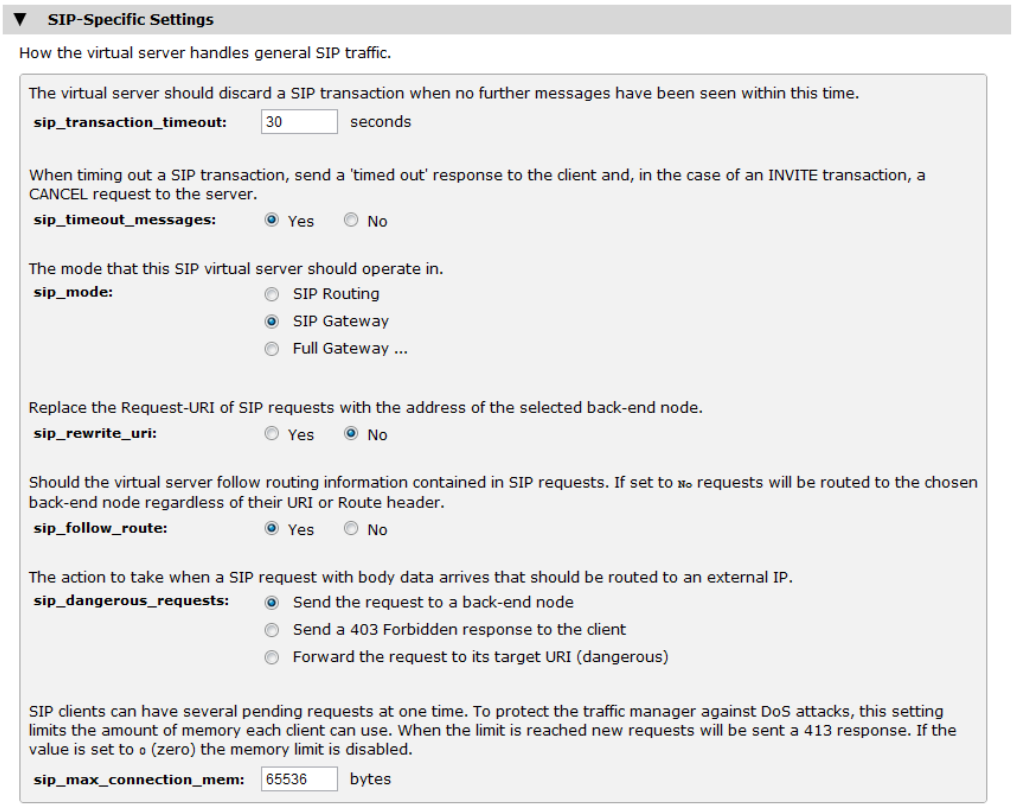


Fig. 27. SIP virtual server settings

The different modes of operation are described below:

- **SIP Routing**: select this option to load-balance SIP sessions between your SIP proxy servers. The first SIP request and all responses to it will pass through the traffic manager and can be inspected and manipulated by RuleBuilder and TrafficScript rules.

  The traffic manager will not add a Record-Route header field to the request. As a result, future requests that are part of the same session will not pass through the traffic manager but will instead go straight to the server that the original request was sent to.

In a typical SIP session, this means the INVITE request and all responses to it will pass through the traffic manager. Any subsequent requests that manage this session, such as BYE requests, will not pass through the traffic manager.

- **SIP Gateway**: select this option to inspect every SIP request that is part of a session. When a request passes through the traffic manager, the traffic manager will add a Record-Route header field to it so that future requests that are part of this session will be routed through the traffic manager. All of these requests and the corresponding responses can be manipulated with RuleBuilder and TrafficScript rules.

  In a typical SIP session, all the control messages such as INVITE and BYE will pass through the traffic manager, but the session data itself will not. This is the default mode of operation for SIP virtual servers in the traffic manager.

- **Full Gateway**: select this option to make all session data pass through the traffic manager. This mode is useful if all your SIP clients are on the same network as your traffic manager and the traffic manager is acting as a gateway to the internet.

  In this mode, all control messages such as INVITE and BYE will pass through the traffic manager. The traffic manager will modify any Session Description Protocol information contained in the body data of a request or response so that the session data also passes through the traffic manager.

  You can choose to specify a port range that will be used for the session data. You can also specify how long the traffic manager will wait before closing the session data connection when no data is being sent over it.

Additional SIP settings

The connection management page also offers some additional settings that affect the behavior of the traffic manager. These settings are:

- **sip_transaction_timeout**: SIP requests and responses are grouped into transactions. There are two types of transactions in SIP: INVITE transactions and Non-INVITE transactions. An INVITE transaction consists of an INVITE request, followed by a series of responses. If the final response was not a '200 OK' response then it is followed by an

ACK request. A Non-INVITE transaction consists of a request followed by zero or more responses. A client can send several different transactions over the same connection.

When no further messages that are part of a particular transaction have been seen for the time specified in **sip_transaction_timeout**, the traffic manager will reclaim the resources associated with the transaction.

If the transaction is not complete when it times out, the traffic manager will issue a 408 request timeout response to the client.

- **sip_rewrite_uri**: If set, this option tells the traffic manager to replace the URI of incoming requests with that of the back-end node the traffic manager has selected. For example, if a request arrives with the URI 'sip:bob@example.com' and the traffic manager is responsible for the domain example.com, then the request that is sent to the back-end node will be sip:bob@10.100.1.2:5070 if the back-end node's IP is 10.100.1.2 and its SIP server is running on port 5070.

  This setting is useful if your SIP proxies do not check the domain of the user when looking up their location.

- **sip_follow_route**: If set, this option tells Stingray to follow the routing information contained within incoming requests. If there is a Route header the request will be sent to the corresponding destination. If there is no Route header, the request will be sent to the destination specified by the URI. If the URI points to Stingray then the request will be sent to the pool as normal.

  This behaviour can be overridden by the **sip_dangerous_requests** setting. For example, if a request arrives that contains body data and a Route header, the Route will be ignored and the request sent to the pool if **sip_dangerous_requests** is set to send dangerous requests to a back-end node.

  If **sip_follow_route** is set to No then all requests will be sent to the pool and will have a Route header added that corresponds to the chosen node.

- **sip_dangerous_requests**: SIP requests contain their own routing information, and as such it is possible for them to be used to attack a

remote computer through an intermediate machine. The traffic manager considers a request dangerous if its next destination is an external IP and the request contains body data.

The traffic manager can handle potentially dangerous requests in several ways:

1) It can send the request to a back-end node. This option is useful if your SIP proxy servers are responsible for domains that do not resolve to the traffic manager's IP. This is the default setting for potentially dangerous requests.

2) It can send a 403 Forbidden response back to the client. This option is useful if you want to reject any requests that are not addressed to your traffic manager's domain.

3) It can forward the request to its target URI**.** Selecting this option means SIP requests will be routed normally. Use this setting when you want your Virtual Server to handle outbound SIP traffic.

- **sip_max_connection_mem**: A client can send multiple SIP requests on the same connection and receive responses to them in any order. To stop a client from having an excessive number of active requests awaiting a response, Stingray can limit the maximum amount of memory each connection can allocate.

  This setting specifies how much memory Stingray should allow each connection to use before refusing new requests with a '413 Request Entity Too Large' response. If this value is set to 0 then the memory limitations are removed, however this will leave Stingray more vulnerable to a Denial of Service attack.

Communicating with UDP-based SIP servers

Typically, UDP-based SIP servers respond to requests much like TCP-based SIP servers – by sending the response from the same port on which they received the original request. This behavior is not explicitly required, however, and some SIP servers might respond from a different port or even from a different IP to the one on which it received the corresponding request.

Stingray caters for this behavior with a configuration setting ‘udp_accept_from’ on the **Pools > Edit > Connection Management** page (where the pool is used by a UDP

service). Here you can select how the traffic manager receives responses to UDP requests, from the following options:

- Only the IP address and port to which the request was sent;
- Only the IP address it was sent to but any port;
- A specific set of IP addresses but any port;
- Any IP address and any port.

These settings provide a varying degree of trade-off between security and compatibility. Allowing responses from multiple IPs can pose a greater security risk due to the increased possibility of fraudulent responses, yet certain SIP servers may require this functionality to communicate correctly. You should select the option that most closely matches your requirements.

If you choose to only accept responses from a specific set of IP addresses, you will need to enter a CIDR Mask (such as 10.100.0.0/16) in the box provided.

Important: Please note that requests sent to an IPv4 address will expect a response back from an IPv4 address only. Conversely, requests sent to an IPv6 address will allow a response back in either IPv4 or IPv6 form.

An associated config setting is provided for SIP (over UDP) Health Monitors. See the *Built-in Health Monitors* section of CHAPTER 14 for further details.

CHAPTER 12 Session Persistence

This chapter explains how the traffic manager is used to provide persistent sessions (also known as sticky sessions) between clients and back-end servers.

What is Session Persistence?

Many classes of requests from clients can be load-balanced across a pool of back-end servers. Multiple requests from one client can be shared across the back-ends with no disruption to service. However, there are certain exceptions, such as server applications which depend upon storing information about a client locally, which may not readily be load-balanced in this way. These include:

Web mail applications that track a user's login by storing files on disk;

Shopping carts that store the contents of a client's cart on disk or in memory;

Programs or protocols with a higher start-up time than normal (such as Java applications or SSL connections).

In these situations, the traffic manager can ensure that requests are mapped to the same back-end server for each request, for the duration of the client's session:

When the traffic manager receives a new connection, it uses its load balancing logic to choose a node for that connection. The traffic manager then records the chosen node in a session persistence map.

When another connection in the same session is received, the traffic manager uses the node that was chosen previously.

In this way, all connections in the same 'session' are pinned to the same back-end node. The session persistence class in use defines how a session is identified, and you can refine this decision using several TrafficScript methods.

Session persistence classes can be used to direct all requests in a client session to the same node. This may be necessary for complex applications, where an application session may be maintained over a number of separate connections. Examples of this include web-based shopping carts, and many complex UDP-based protocols.

Session persistence should only be used when necessary. It effectively bypasses the load-balancing process for all but new sessions, so unless it is necessary to send all

requests from the same client to the same back-end node, response times will be better without session persistence.

Configuring Session Persistence

Session persistence is configured via the Catalog. Session Persistence Classes are added to the catalog, and can then be assigned to a Pool, or selected in a TrafficScript rule.

When TrafficScript is not being used, Session Persistence is enabled on a *per-pool* basis. You can define multiple pools, and enable session persistence only for those that require it. Note that a back-end server can be in more than one pool.

Several pools can refer to the same session persistence class to share session persistence mappings. For example, a pool containing some HTTP nodes (on port 80) and a pool containing the same nodes running HTTPS services (on port 443) can share the session persistence mappings for users to the services.

When TrafficScript is being used, Session persistence can be overridden with a TrafficScript command. The traffic manager will use the Session Persistence Class specified for the pool which handles the request unless this command is used.

Enabling Session Persistence

First, click the **Catalogs** button, then click on the **Persistence** tab to open a list of Session Persistence Classes managed by the traffic manager. Next, enter a name in the box labeled **Create new Session Persistence class** and click **Create Class**.

The user interface now shows a summary of the Session Persistence Class settings.

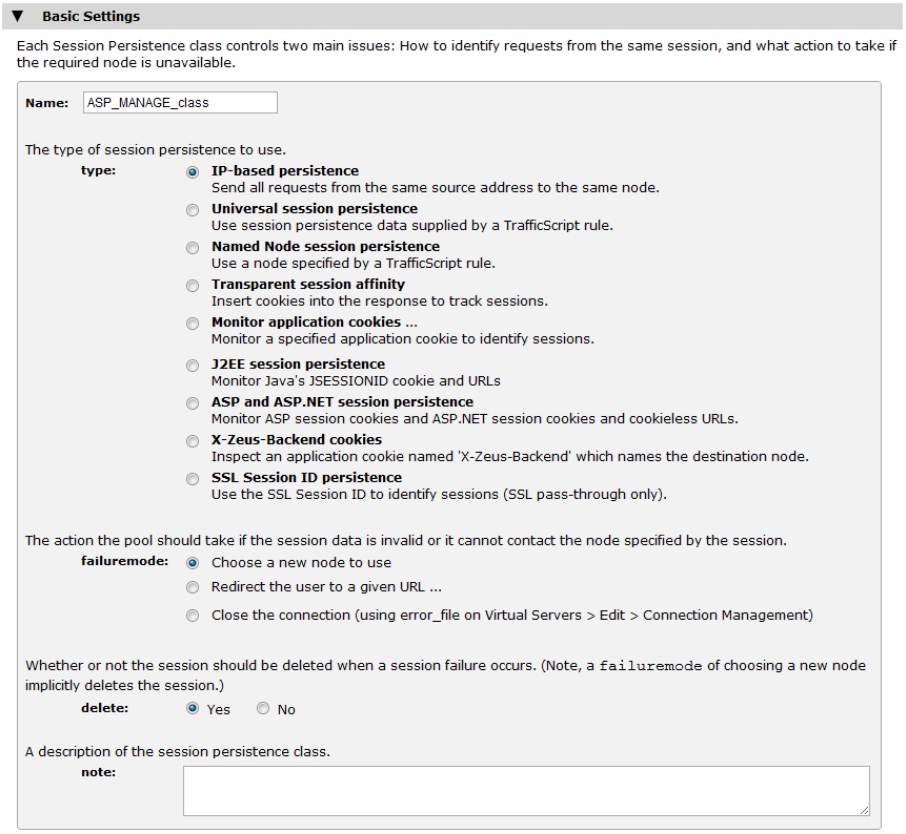


Fig. 28. Session persistence's basic settings

Selecting a Persistence Method

By default, new Session Persistence Classes are configured with **IP-based Persistence** selected. The following alternatives are available when session persistence is required:

Method	Applicable Protocols	Description
IP-based persistence	All	Send all requests from the same source address to the same node.
Universal session persistence	All	This class allows you to persist sessions based on any information in a request by setting a unique persistence key in a TrafficScript rule.
Named Node persistence	All	With this method, a TrafficScript rule can direct the connection to a specific node.

Transparent session affinity	HTTP, HTTPS	Insert cookies into the response to track sessions.
Monitor application cookies	HTTP, HTTPS	Monitor a specified application cookie to identify sessions.
J2EE	HTTP HTTPS	Monitor Java's JSESSIONID cookie and its URL-rewriting fallback.
ASP and **ASP.net**	HTTP, HTTPS	This method allows both types of ASP sessions to be processed using session persistence.
X-Zeus-Backend cookies	HTTP, HTTPS	Inspect an application cookie named `X-Zeus-Backend` which names the destination node.
SSL Session ID persistence	SSL protocols	Use the SSL Session ID to identify sessions.

IP-Based Persistence

When IP-based persistence is selected, the traffic manager will track the originating IP address for each request to this pool. If the traffic manager has already received traffic from this address, it will map requests to the same back-end server it used previously.

Some specialized mobile clients may change their IP address during a session, and some clients may share IP addresses (for example, via a proxy server), so this may not be suitable for some environments.

However, since IP-based persistence is completely transparent and is not protocol-dependent, it is useful for guaranteeing persistent connections for difficult protocols or applications.

IP session maps are shared by all traffic manager machines in a cluster. Requests received by different traffic manager machines will be directed to the correct node, and if one traffic manager fails, the other traffic managers are aware of the IP session maps it was maintaining.

Universal Persistence

Universal session persistence uses a persistence key, defined in a TrafficScript rule, and seeks to direct all connections with the same persistence key to the same back-end node.

The persistence key is specified in a rule by the `connection.setPersistenceKey()` TrafficScript function. This allows persistent sessions based on any information in a request; see the *Universal PHP Persistence* section of CHAPTER 12 for an example.

Universal session persistence maps are shared by all traffic manager machines in a cluster. Requests received by different traffic manager machines will be directed to the correct node, and if one traffic manager fails, the other traffic managers are aware of the universal session maps it was maintaining.

Important: Note that when using Universal Session Persistence with the SIP protocol over UDP, session persistence will only be applied to the first request sent by a client. All subsequent requests from the same client will be sent to the same node as the first request, even if they have been assigned a different session persistence key. Universal Session Persistence should therefore be used with SIP only when you want all requests from the same client to be directed to the same server.

Named Node persistence

Named Node session persistence can be used to direct the connection to a specific node in a pool. It gives very fine-grained control over how requests are routed.

A TrafficScript rule can use the `connection.setPersistenceNode()` function to specify precisely which node should be used in the session persistence decision. For example, when a new connection for a Remote Desktop service is received, a rule could query an external database to determine the node which hosts that user's desktop, and then use the `connection.setPersistenceNode()` function to direct the connection to that node.

Transparent Session Affinity

Transparent session affinity inserts cookies into the HTTP response to track sessions. This is generally the most appropriate method for HTTP and SSL-decrypted HTTPS traffic, because it does not require the nodes to set any cookies in their response.

Transparent session maps are stored in a client-side cookie named **X-Mapping-xxxxxx**, where 'xxxxxx' is an opaque string that identifies the session persistence class and the value of the cookie is an opaque string that identifies the preferred node. All traffic manager machines in a cluster will inspect the value of this session cookie and send the session to the same server node.

Monitor Application Cookies

Application cookie persistence monitors a named cookie in the HTTP response from the node. For example, PHP applications may generate session cookies named "PHPSESSID" that the clustered PHP application could track and index session state. This persistence method directs a request to the same server node if it contains an application cookie. You need to specify the name of the application cookie you wish to monitor.

Application cookie session maps are stored in a client-side cookie **K-CookieName-xxxxxx**, where CookieName is the name of the monitored cookie, 'xxxxxx' is an opaque string that identifies the session persistence class and the value of the cookie is an opaque string that identifies the preferred node. All traffic manager machines in a cluster will inspect the value of this session cookie and send the session to the same server node.

If the back-end server changes the value of the application cookie, the session is still valid and clients will continue to be directed to the same server node.

J2EE JSESSIONID cookies/URL

J2EE JSESSIONID cookies/URL persistence monitors both the JSESSIONID cookie (as for application cookie persistence) and the jsessionid path parameter. These are defined in the Java extension specification (v2.4) and are used by Java extension containers such as BEA WebLogic, IBM WebSphere Application Server, JBoss/Tomcat and others.

J2EE session maps are shared by all traffic manager machines in a cluster, so requests received by different traffic manager machines will apply the same sessions.

ASP.net Session Persistence

ASP (Active Server Pages) is a server-side scripting protocol created by Microsoft to handle dynamically generated web pages. In order to use ASP Session management in a server cluster, the *same* Web server must handle all requests coming from a user for the life of the session.

ASP sessions can use cookies or can be cookieless, depending on numerous factors, such as the lack of support of cookies by the browser or the user disabling them voluntarily.

- **For cookie-based sessions:** The traffic manager's ASP Session Persistence class detects and uses the cookie to identify the client's session.
- **For cookieless sessions:** The traffic manager's ASP Session Persistence class detects and uses the ASP identifier embedded in URLs generated by the ASP application.

ASP session maps are shared by all traffic manager machines in a cluster, so requests received by different traffic manager machines will apply the same sessions.

Note that if you have several distinct ASP applications hosted behind a single hostname, each application will generate its own cookie. You will need to have a separate ASP.Net session persistence class for each application so that mappings between cookies and nodes are managed independently for each application. For example, you may inspect the request URL to determine which application is being accessed and select the session persistence class using the TrafficScript function `connection.setPersistence()`.

X-Zeus-Backend Cookies

X-Zeus-Backend cookie persistence looks for a cookie named **X-Zeus-Backend** in each application request. If the cookie is present, and contains the name of a node in the current pool, the request is sent to that node. The cookie can be inserted either by a back-end server or by a TrafficScript rule.

Note: This persistence method is deprecated, and provided only for backward compatibility with Zeus Load Balancer persistence cookies.

SSL Session ID Persistence (SSL pass-through only)

The SSL session ID persistence method sends all SSL traffic with the same SSL session ID to the same server node. It is only applicable to SSL pass-through traffic, not SSL-decrypted traffic.

The SSL session ID persistence method reduces the number of SSL handshake operations your nodes perform. SSL handshakes are expensive in terms of CPU time, network bandwidth and latency.

SSL Session ID session maps are shared by all traffic manager machines in a cluster. Requests received by different traffic manager machines will be directed to the correct node, and if one traffic manager fails, the other traffic managers are aware of the SSL Session ID session maps it was maintaining.

> SSL session ID persistence is not appropriate for application-level session persistence because many SSL clients regularly renegotiate their SSL session

ID. To achieve application-level session persistence you should either use IP-based session persistence, or decrypt the traffic and use universal session persistence or an HTTP method if applicable.

Resolving session persistence maps to nodes

When the traffic manager receives a new connection and there is no session persistence information, the traffic manager uses its load balancing logic to choose a node for that connection. The chosen node (IP address and port) is recorded in the session persistence mapping, which is either internal (in the case of IP, Universal or SSL session persistence) or in an external cookie (Transparent, Application cookie or X-Zeus-Backend cookie).

When another connection in that session is received and the traffic manager is ready to forward that connection to a node in a pool, the traffic manager inspects the session persistence class in use to determine if a particular node should be used.

If a node with the same IP address and port exists in the pool for the connection, then the traffic manager sends the connection to that node.

If there is not an exact match, the traffic manager searches the pool for any nodes with the same IP address (but different ports). If just one such node exists, the traffic manager sends the connection to that node.

This allows session persistence information to be shared between different pools with different nodes types.

For example, a web-based application may use an HTTP interface to manage items in a shopping cart, and a secure HTTPS interface to manage payment. Session persistence requirements may dictate that users must be directed to the same physical machine for both HTTP and HTTPS traffic.

Both services (HTTP and HTTPS) could reference the same session persistence class. When a user first connects to the HTTP service, the traffic manager would use the HTTP pool. A session would be established with a particular node (IP address, port 80) in that pool.

When the user accesses the HTTPS service, the traffic manager might use a different pool containing HTTPS nodes. The session persistence class will then direct his request to the same physical machine (IP address, port 443) in the HTTPS pool.

Node Failure Options

Sometimes, the node required by the session persistence mapping may not be available. For example, it may be marked as 'failed' by a monitor, or the traffic manager may be unable to connect to it.

The traffic manager provides alternative actions for sessions currently using that node. At the bottom of the Session Persistence page are radio buttons to select the action:

Choose a New Node to Use

The pool discards the session map and chooses a new node, using the current load-balancing algorithm.

Close the Connection

The pool immediately closes the connection. HTTP traffic will send an error file of pools.

You can choose whether to discard or remember the session map. If you remember it, then if the client returns and the session's node is available again, the request will be sent to the session's node.

Send an HTTP Redirect (HTTP only)

The pool sends an HTTP 302 redirect to the configured location (URL) as a response to the request. The resource at the redirect location could display a message, or cause the user to log in again and establish a new session.

You can choose whether to discard or remember the session map. If you remember it, then if the client returns and the session's node is available, the request will be sent to the session's node.

Draining Connections

Sometimes it is necessary to take a back-end server out of service; for example, to upgrade software, perform hardware maintenance or to decommission or repurpose it.

A pool's Connection Draining capability is designed to facilitate this. When you mark a node as 'draining', the traffic manager stops sending it any new connections. However, any connections that are in a session previously established to that node are still sent to that node.

This allows you to safely remove a node from a pool without interrupting either ongoing connections, or longer-term established sessions. The **Activity** section of the Admin Server provides reports so that you can discover how long a draining node has been idle, and then make a judgment as to whether all sessions have completed.

For example, suppose you have an e-commerce service and you use session persistence to tie individuals' sessions to particular back-end nodes. You mark one of your nodes as draining.

No new sessions are established with that node.

Existing, established sessions are allowed to continue with that node.

After 60 minutes, you inspect the **Draining Nodes** page in the **Activity** section of the Admin Server. You observe that the node has been idle for 35 minutes. It is probably safe to conclude that all established sessions have now completed, and you can remove the node safely.

In practice, the time periods involved in determining whether a node has finished draining will be very much dependent on the application and user behavior. In setting up the system, an administrator will have to decide these values in relation to the desired use.

The *Draining and Disabling Nodes* section of CHAPTER 5 describes how connection draining is configured in a Pool.

Sizing the session persistence caches

Some session persistence methods use client-side cookies to store the session persistence data. The remaining session persistence methods use caches in the traffic manager, shared automatically across a cluster, to store session persistence mapping.

The caches are fixed in size. When a cache fills up, the oldest (least-recently-used) entry is discarded when a new entry is added. Cache sizes are configured in the **System > Global Settings** page, in the **Cache Settings** section:

IP session persistence	**ip_cache_size**
Universal session persistence	**universal_cache_size**
SSL session-id session persistence	**ssl_cache_size**
J2EE session persistence	**j2ee_cache_size**
ASP session persistence	**asp_cache_size**

You can monitor the behaviour of the caches using the Activity Monitor. The key values to help you size the cache are:

Entries: The number of entries in the cache

EntriesMax: The configured maximum size of the cache

Oldest: The time since the least-recently-used entry was last used

Once the traffic manager has processed a number of sessions, it is normal for the cache to completely fill up. The **Oldest** value will indicate the current session expiry time – how long entries are retained without being used before they are discarded.

To help you size your cache, you need to consider the rate at which new entries are added to your cache (entries per second) and the length of time (in seconds) that you want these entries to be retained since they were last used (the session expiry time). Multiply these two values together to get an estimate of the required cache size, and monitor the **Oldest** value to check that entries are not prematurely discarded because the cache has filled.

If you need fine-grained control of session persistence records, the most effective means is to use client-side cookies with specific expiry times, and tie the session persistence to the presence of the cookie.

Using Session Persistence with Multi-hosted Traffic IP Addresses

If you use multi-hosted Traffic IP Addresses with the 'consider source port' setting enabled, then requests from one source IP address (i.e. from a client) will be handled by all of the traffic managers in your cluster (see the *Creating a Traffic IP Group* section of CHAPTER 6).

In this situation, session persistence methods that rely on state sharing (session maps are shared between the traffic managers) may not work reliably. There is a short delay before session information is propagated across the cluster, and clients may visit several traffic managers during this period, resulting in corruption of the client's session persistence mapping.

Session persistence algorithms that depend on state sharing are:

IP-based Session Persistence

Universal Session Persistence

SSL SessionID Session Persistence

J2EE and ASP.NET Session Persistence

Do not use the 'consider source port' setting in any multi-hosted Traffic IPs that are used by services that use any of the above session persistence methods.

Session Persistence with UDP protocols

UDP protocols are not connection oriented, but you will often desire that UDP datagrams in the same session are routed to the same back-end server.

In the **Virtual Server > Connection Management** settings, the two settings **udp_timeout** and **udp_response_datagrams_expected** are used to inform the traffic manager what the UDP session should look like:

The UDP session will last until the number of response datagrams in **udp_response_datagrams_expected** have been observed. For example, if a session completes once the server has send one datagram to the client, set this value to '1'

Set the value to '-1' if the session is long-lived.

The UDP session will time out and if no further UDP traffic has been observed within **udp_timeout** seconds.

You can use Session Persistence of various types (typically IP-based or Universal) with UDP if you wish to track sessions for longer periods of time.

Examples

Universal PHP Persistence

A TrafficScript rule can inspect incoming HTTP requests, and select a back-end node based on the request.

The example in the *Routing by Content Type* section of CHAPTER 8 gives the TrafficScript rule to select a pool based on filename extension. This rule can be modified to pass session persistence data to a chosen Session Persistence Class.

Create a Session Persistence Classes called `PHP`, and set to use universal session persistence (see the *Configuring Session Persistence* section of CHAPTER 12). Next create a TrafficScript rule in the catalog as follows:

```
$path = http.getPath();
if( string.endsWith( $path, ".php" )) {

  # Persist on the PHPSESSID cookie, IP
address and
  # user agent
  $phpCookie = http.cookie( "PHPSESSID" ) .
               connection.getRemoteIP() .
               http.getHeader( "User-Agent"
);

  # Set the persistence key to our unique
value
  conection.setPersistenceKey( $phpCookie
);

  # Select the PHP Session Persistence
Class
  connection.setPersistence( "PHP" );
}
```

Apply this rule to the virtual server handling your traffic. The string passed to the `pool.use()` function (`$phpCookie`) is the session persistence data used by the pool.

Requests for files with other extensions (such as `.html` or `.jpg`) are ignored by the above rule. These requests are passed on to the other rules used by the virtual server, or to its default Session Persistence Class as set in the default Pool.

Also note that only one Pool is required, and the Session Persistence is managed entirely by the two separate Session Persistence Classes.

CHAPTER 13 SSL Encryption

This chapter explains how to use Secure Sockets Layer (SSL) encryption with the traffic manager. It includes a description of SSL and how to use the traffic manager to manage authentication and encryption, as well as the storage of certificates.

Overview of SSL

SSL (Secure Sockets Layer) is a protocol used to send traffic securely over the Internet. Traffic is encrypted using a key agreed between the server and client machines.

SSL provides several advantages:

Server authentication

Client authentication

Encrypted data transfer

SSL can be used with almost any TCP/IP protocol, but is most commonly used to secure HTTP (web) traffic, forming the HTTPS protocol.

Server Authentication

A server identifies itself for SSL communications using an ***SSL certificate***. This certificate contains the name and location of the organization and its DNS name, and gives the client assurance that they are accessing the correct site.

An SSL certificate can be ***self-signed*** by the organization that owns it. However, without independent verification, this certificate will not automatically be trusted by a client. To be trusted, it must be signed by a recognized, independent ***certificate authority (CA)*** such as Verisign or Thawte. The organization sends a ***certificate signing request (CSR)*** to the CA, which carries out thorough checks on the details in the certificate, and may also inspect the organization's financial records. Note that certificate authorities charge for this service.

Client Authentication

In some cases, you may wish to only allow certain approved people to access your service, for instance a company intranet or extranet. To achieve this, you can require that the client provides an SSL certificate signed by a trusted certificate authority.

The traffic manager supports two methods to ensure the validity of these certificates, ***Certificate Revocation Lists (CRLs)*** and the ***Online Certificate Status Protocol (OCSP)***:

Certificate Revocation Lists

Within the traffic manager your trusted certificate authorities are held in a catalog. Each certificate authority can distribute ***certificate revocation lists (CRLs)***, which are also held in this catalog. Certificates usually have a fixed validity period, such as 12 months, but sometimes a certificate is cancelled before it expires. In this case the certificate authority adds it to a certificate revocation list, so that it will no longer be trusted.

Online Certificate Status Protocol

OCSP is an internet protocol used for obtaining the current validity of an SSL client certificate at the point of use. It was created as an alternative to CRLs in order to address some of the inherent shortcomings of that method, such as the limitation that updates must be frequently downloaded to keep the list current.

When a user attempts to access a secure service, OCSP sends an HTTP request to an OCSP server (known as a *Responder*) for the certificate's status information. This request is packaged up in the form of an *ASN.1* message, optionally signed with a certificate, and sent to the responder. In return, the responder sends back a response of "current", "expired" or "unknown". Unless the certificate is reported as being current, the SSL connection is terminated. The *Configuring OCSP* section of CHAPTER 13 provides full details of how to configure OCSP for your secure services.

Further information on OCSP can be found at http://tools.ietf.org/html/rfc2560.

Encrypted Data Transfer

Once server and client are satisfied with each other's identity, they agree on an encryption key to use for data transfer. This is different from their identification keys for reasons of efficiency. Data is encrypted before transfer so that a third party cannot read it. In addition, SSL has reliability features which ensure that any disruption to the data stream is detected. These features give client and server confidence that their communication is private, and has not been corrupted.

SSL Features in Stingray traffic manager

Decryption and Encryption

Stingray can decrypt SSL traffic within the virtual server. This can be useful for two reasons:

After decryption, the traffic manager's traffic analysis features can be used on the whole request. Service protection methods can filter for malicious content, viruses or web worms; and rules can inspect the headers and body of the request to make an informed routing decision. Without decrypting the packets very little information is available.

Decryption requires processing power. It may be more efficient if the traffic manager decrypts requests before passing them on to the nodes, reducing the load on the back-end servers.

If your virtual server is decrypting SSL traffic in order to use rules, you may wish to encrypt it again before sending it to the nodes. This provides complete end-to-end security in your system. SSL encryption is handled by the pools in the traffic manager.

SSL Certificates Catalog

The traffic manager provides a catalog store, containing sets of objects which your traffic managers can use when handling services.

Using catalogs, the traffic manager provides a centralized store of SSL server certificates, client certificates, certificate authorities and certificate revocation lists.

SSL Decryption Wizard

The SSL Decryption Wizard provides a simple, step-by-step process to correctly configure SSL Decryption. The wizard performs the following steps in a simple, walk-through interface:

Enables SSL decryption for an SSL Virtual Server

Assigns an SSL certificate from the Catalog to the Virtual Server

Enables SSL encryption for the default Pool used by the Virtual Server

Changes all protocol types to the underlying (non-SSL) protocol type.

Note: Prerequisites: to use the SSL Decryption Wizard, you must have at least one SSL Virtual Server (of any type). If you do not have the necessary SSL certificates, the traffic manager will help you create a certificate from the wizard.

Adjacent to the help icon, pull down the **Wizards** menu and choose **SSL Decrypt a service**. A new window opens which explains that the wizard will configure the

service to be decrypted on receipt, and re-encrypted before being passed to a pool. Click **Next**.

Select the service which you wish to decrypt. Note – only Virtual Servers that use an SSL protocol are listed. Click **Next**.

Now choose which certificate will be used to decrypt the incoming requests (or click **Create New** to add a certificate now). Click **Next**.

On the next page, choose a protocol type. This is the underlying decrypted protocol. For instance, HTTPS requests are decrypted to HTTP internally. Simply choose the underlying protocol, HTTP, as this matches the protocol type you are accepting, without SSL wrapping.

When you have made your selections, click **Finish** to complete the wizard.

Configuring SSL Certificates

The following configuration options are offered from the **Catalogs > SSL** page on the Admin Server. Note that the following sections refer to functionality applicable to both Client and Server SSL certificates:

- *Server SSL Certificates* are used to identify SSL-encrypted services hosted by the traffic manager.
- *Client SSL Certificates* are used when the Traffic Manager needs to authenticate itself against an SSL node.

Creating a new self-signed SSL Certificate

First, click the **Catalogs** button on the top bar of the Admin Server interface. Click the **SSL** tab. From here, click on the **Edit** button for either **SSL Certificates Catalog** (for Server certificates) or **SSL Client Certificates catalog** depending on your requirements.

The traffic manager lists any existing certificates that have been configured, and allows you to create or import a certificate. For testing purposes and internal use, an SSL certificate can be ***self-signed***. This means that the certificate has not been signed by a trusted third party and may not be relied upon as a means of authenticating the server.

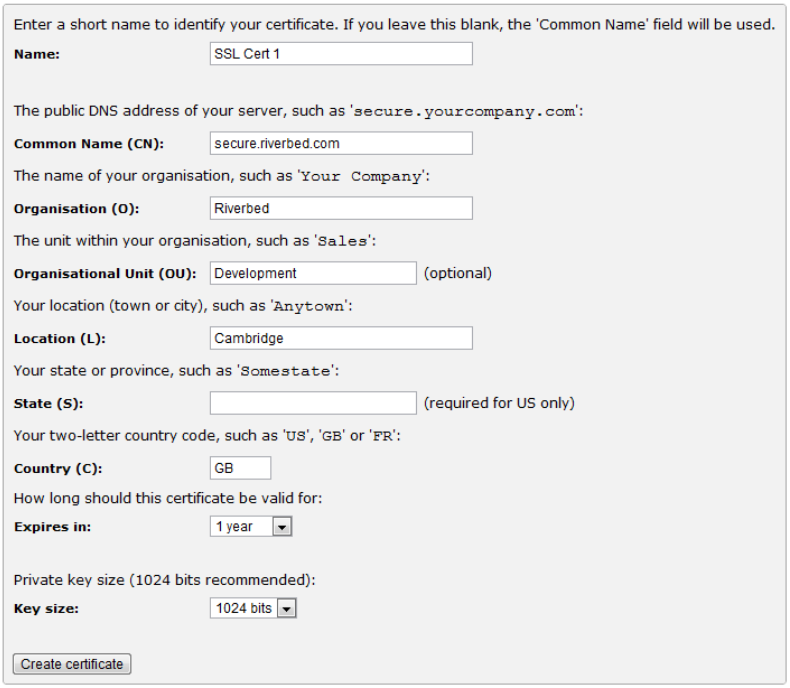


Fig. 29. Main settings for a self-signed certificate

Click on **Create Self-Signed Certificate / Certificate Signing Request**, and complete the form.

Give the certificate a **name** and a **Common Name** (CN), which should be the DNS name of the server that will use this certificate, such as `secure.yourcompany.com`. Most clients will show a warning message if the CN in the certificate and the DNS address of the server do not match.

Supply values for your **organization, organizational unit** and **address.** These will appear in the certificate.

When you enter your country code, use the **two-letter ISO country code**[7]. For example, Great Britain is "GB", and Germany is "DE".

Set an **expiry date** for the certificate (default, one year) and **a private key size.** Please note that while 2048-bit keys provide a higher level of security, they require more processing power and are not supported in all clients.

Now click **Create Certificate**. If there are no errors the traffic manager will create an SSL certificate and place this in the SSL Certificates Catalog.

[7] As defined in the ISO-3166 standard.

You can verify that the certificate has been added by clicking the expansion tab on the **Catalogs** page, beside **SSL Certificates Catalog**. Each certificate is listed in a table, and the expiry date of the certificate is also indicated.

Managing Certificate data

You can change any of the values stored within a self-signed certificate. To do this, click on an existing SSL certificate in the appropriate Client or Server SSL Certificates Catalog.

In the **Edit Certificate** box, change any of the existing values for the certificate and then click **Update Certificate**.

Note: You cannot edit a certificate signed by a Certificate Authority, as this would invalidate the signature.

You can also copy an existing SSL certificate, and assign a new certificate name to the copy. To do this, click on an existing certificate in the SSL Certificates Catalog. You may need to expand the Client or Server SSL Certificates Catalog first, by clicking on the expansion tab.

In the **Copy Certificate** box, enter a new name for the copy and click **Copy Certificate**. The new SSL certificate is added to the relevant catalog immediately.

Creating a Certificate Signing Request

A Certificate Signing Request (CSR) is a formal request made by an individual to a certificate authority (CA) to obtain a digital identity certificate. Certificate Authorities are entities responsible for issuing certificates for use by other parties.

Once you have created your self-signed certificate, you can then create a Certificate Signing Request based on the information in the self-signed certificate:

- Go to the **Catalogs > SSL > [Server|Client] Certs** screen;
- Click to edit the required certificate;
- In the **Certificate Signing** section, click **Export CSR/Sign certificate**;
- The window will now show the text for your certificate request:

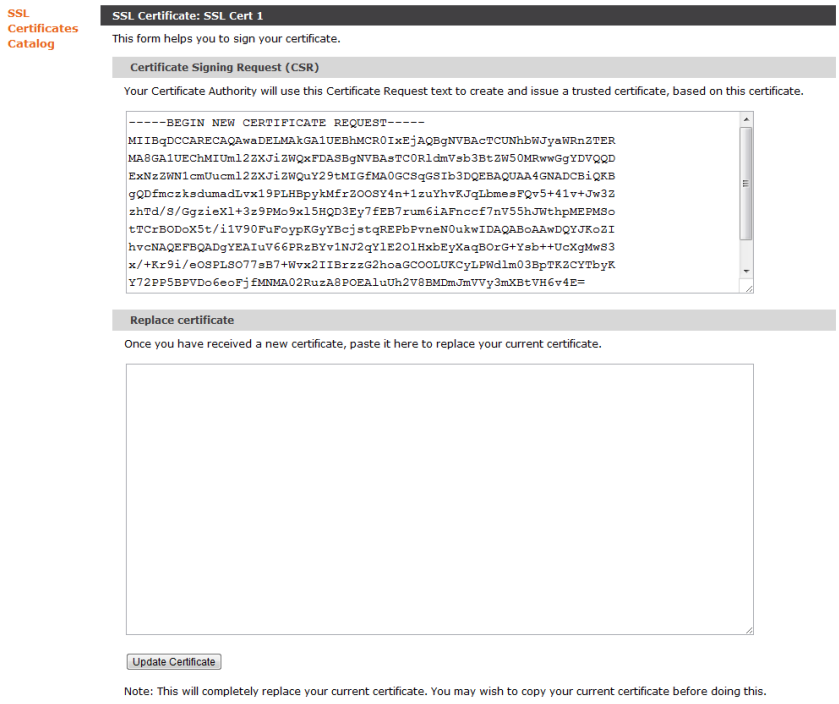


Fig. 30. Your certificate request to the Certificate Authority (CA)

- The text in the upper part of the window contains an encoding of the information in the stored certificate suitable for the CA to sign. You can copy and paste this text into the request;
- You may be asked to attach other credentials or extra information to your request.

The Certificate Authority will return a replacement certificate which they have digitally signed with one of their public certificates (or a certificate in a public certificate chain). Paste the textual contents of the returned certificate in the lower part of the window, and click **Update certificate** to complete the process.

Importing a new SSL Certificate

To import an existing certificate, you will require:

The certificate file

The private key file

These files are in a standard, ***PEM-encoded*** format which can be cut and pasted into a text file for uploading. PEM-encoded certificates are formatted as follows:

```
-----BEGIN CERTIFICATE-----
MIIBPLMAkGA1UEBhMCWkExFTATBgNVBAgTDFdlc3Rlcm4gQ2Fw
ZTESMwZSBUb3duMRQwEgYDVQQKEwtPcHBvcnR1bml0aTEYMBYG
A1UEClcnZpY2VzMRowGAYDVQQDExF3d3cuZm9yd2FyZC5jby56
YTBaMBAQUAA0kAMEYCQQDT5oxxeBWu5WLHD/G4BJ+PobiC9d7S
6pDvAtXdm2j190D1kgDoSp5ZyGSgwJh2V7diuuPlHDAgEDoAAw
DQYJKDQQBf8ZHIu4H8ik2vZQngXh8v+iGnAXD1AvUjuDPCWzFu
pReiq7UR8Z0wiJBeaqiuvTDnTFMz6oCq6htdH7/tvKhh==
-----END CERTIFICATE-----
```

Click on **Import Certificate** in the relevant **Client/Server SSL Certificate Catalog** to perform the upload. Provide a name for the imported certificate, and the names of the *Certificate* and *Private Key* files. Now click on the **Import Certificate** button to complete the process. The traffic manager will add the certificate to the appropriate catalog and automatically distribute the key data securely to all traffic managers.

Important: Do not forget to delete temporary copies of your SSL private key that you may have created during the import process. The private key is valuable and should never be stored in an insecure location.

When a certificate signed by a Certificate Authority is not signed directly by the **root certificate** it may be necessary to send clients both the signed certificate and the Certificate Authority's intermediate certificate.

Working with Intermediate Certificates

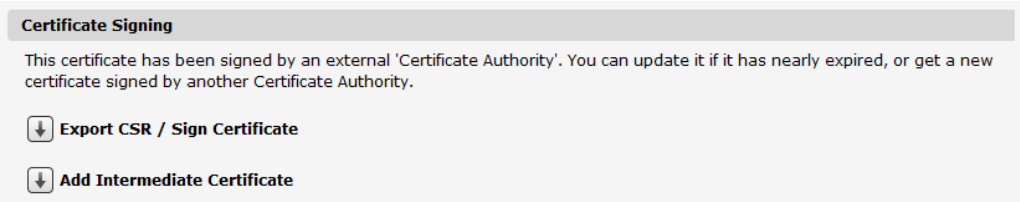
Web Browsers and other SSL clients are preconfigured with a set of **root certificates** from Certificate Authorities that they trust. They will allow the user to connect to an SSL service that uses an SSL Server Certificate signed by one of the trusted Certificate Authorities. Similarly, back-end SSL nodes may be pre-configured to authenticate Client SSL certificates presented by the traffic manager.

However, for ease of management and improved security, many certificate authorities use **certificate chains**. The SSL certificates they distribute are not signed directly by a root certificate; rather they are signed by an intermediate certificate which is itself signed by the root. This forms a chain of trust from the SSL certificate

back to the trusted root certificate. In some cases, a chain of two or more intermediate certificates is used between the SSL certificate and the root.

Web clients and SSL nodes are generally not equipped with the intermediate certificates. If, for example, a web client is presented with a server certificate, it has no way of verifying that it was signed (albeit indirectly) by a trusted root certificate. In this case, the SSL server (i.e. the traffic manager) must present the entire or partial SSL certificate chain so that the client can verify the SSL certificate – the certificate chain includes the SSL certificate and all intermediate certificates. It is not necessary to include the root certificate.

Your certificate authority will inform you if a chain of certificates is used. You need to upload all of the intermediate certificates to the relevant Server or Client SSL certificate catalog:



The traffic manager will verify that the intermediate certificate you upload has signed the current SSL certificate (or the previous intermediate certificate in the chain).

Managing Certificate Authority certificates and CRL Files

The traffic manager can request that remote clients who connect over SSL provide a client certificate to authenticate themselves. You will need to configure the traffic manager with the public certificates from the Certificate Authorities you trust, and optionally any Certificate Revocation Lists (CRLs) that they distribute.

From the **Catalogs** page, click on the **Edit** button for the **Certificate Authorities and Certificate Revocation Lists Catalog**.

The traffic manager can import the CA or CRL file using one of three methods:

A file uploaded via a web browser to the traffic manager;

A URL for a file which the traffic manager will download directly;

A file manually pasted into the **File Contents** box (PEM-encoded).

Select one of the methods and click **Import File**. The traffic manager will then import the CA or CRL file and propagate the new information to all traffic managers.

SSL Decryption

The traffic manager can decrypt and re-encrypt SSL traffic on the fly as it proxies requests between clients and back-end pools; it uses certificates stored in the catalogs, including server and client certificates. After decrypting traffic, perhaps to apply TrafficScript rules, the traffic manager can re-encrypt the data if required before passing it to the nodes. This allows extremely flexible management of requests without compromising security requirements on potentially untrusted networks. The traffic manager thus provides a fully transparent SSL gateway.

Setting up SSL Decryption

SSL decryption is performed by a virtual server. When enabled, SSL decryption will use additional CPU resources in the traffic managers that perform the decryption.

To enable SSL decryption, click **Services** on the top bar of the Admin Server. Click on the **Virtual Servers** tab to open a list of all virtual servers the traffic manager manages. Next, click on the **Edit** button for a virtual server that supports SSL.

In the list of configuration options for the virtual server, click on the **Edit** button for **SSL Decryption**.

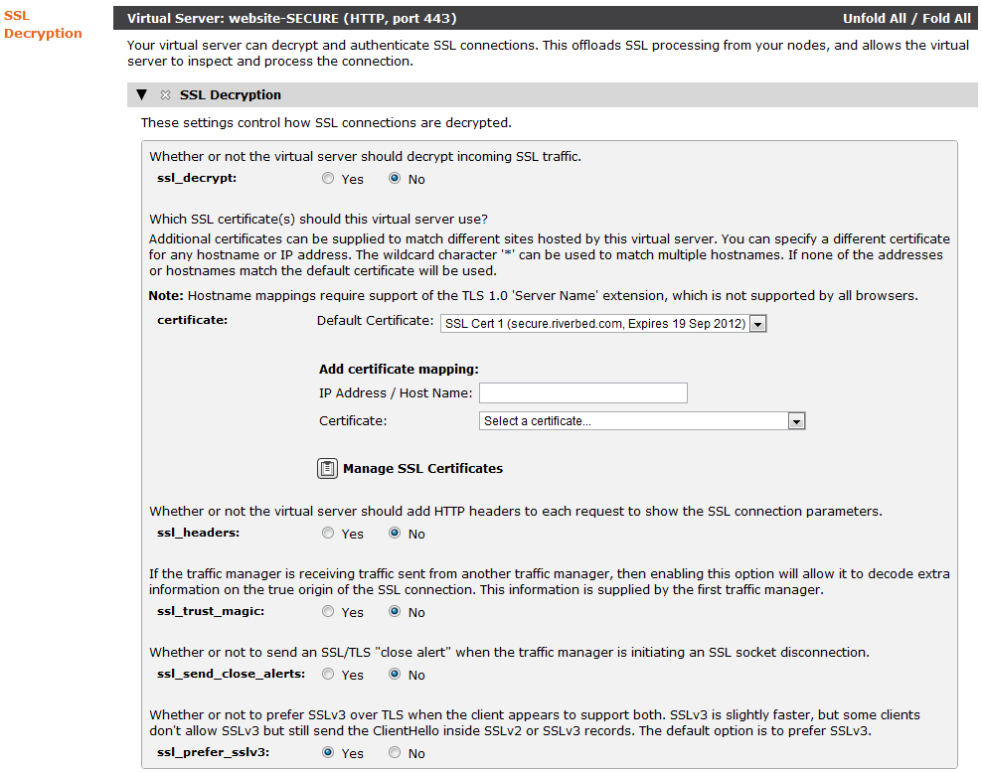


Fig. 31. Setting up a client's SSL decryption and the use of certificates

The traffic manager now displays a list of configuration options for SSL decryption for this virtual server.

Set the virtual server to decrypt SSL traffic using the **ssl_decrypt** radio button.

Select the certificate you wish to use

ssl_headers: The traffic manager includes the option to add HTTP headers with details of the decryption performed (HTTPS decryption only). These describe the client cipher, session ID and whether a valid client certificate was provided, and can be inspected by the back-end application to determine what encryption parameters were used.

Header name	Sample Value
`SSLClientCipher`	`SSL_RSA_WITH_RC4_128_SHA, version=SSLv3, bits=128`
`SSLClientCertStatus`	`NoClientCert` *or* `OK`
`SSLSessionID`	`8723367551317ABE5443278C6E...` (64 hex characters)

- **ssl_send_close_alerts**: It is common for SSL servers to omit the final close alert when shutting down an SSL connection. However, some clients, especially FTP clients, are strict when it comes to handling close alerts and will fail to function properly if the alerts are omitted. This option is disabled by default because it can break some older SSL implementations, including versions of Microsoft Internet Explorer. You can also modify this behaviour on the fly using TrafficScript rules.

Click **Update** to apply the changes to the virtual server.

Using multiple SSL certificates

The SSL certificate contains a value called the Common Name (CN), and a client will generally check that the common name value matches the DNS name it is using to connect to the service. It will warn the end user if they do not match.

If your virtual server is managing traffic for more than one service (e.g. secure.site1.com, secure.site2.com etc), you will want to configure it with multiple certificates and ensure that it sends the correct certificate for the service the client is trying to access.

There are two ways that the traffic manager can distinguish between different services:

Destination IP address: secure.site1.com and secure.site2.com could resolve to different IP addresses. The traffic manager can listen on both IP addresses and select the certificate to used based on the IP address the connection was received on.

TLS Server Name Indication[8]: TLS (an updated version of SSL version 3) has an optional capability where a client can provide the name of the service it is trying to contact (in plaintext) at the very beginning of the TLS handshake. The traffic manager can inspect this name and chose the certificate to use.

Important: The TLS Server Name Indication does not require that each SSL service runs on a dedicated IP address, so it is more scalable and easier to manage. However, at the time of writing, not all common web clients support this feature; clients such as Internet Explorer 7 (Windows XP) will receive the wrong certificate if they attempt to access a service that requires TLS server name support.

Configuring multiple certificates

The traffic manager will use the primary certificate (configured with the **certificate** setting in the Virtual Server) by default. You can override this decision by configuring mappings in the **ssl_sites** setting of the Virtual Server configuration.

You can configure mappings from IP address to certificate; if the client connects to the configured IP address, it will be given the nominated certificate rather than the default one;

You can configure mappings from domain name (or wildcard domain) to certificate; if the client provides a server name in their SSL handshake that matches the domain name, they will be given the nominated certificate.

When the traffic manager receives a request it will check the certificate settings to determine the certificate to send. It will chose the first matching certificate: exact matches of server name come first, then wildcard matches of server name, then exact matches of IP address and finally wildcard matches of IP address. If there are no matches, then the default certificate is used.

---

[8] http://en.wikipedia.org/wiki/Server_Name_Indication

Client Certificates

SSL (server) certificates are used to identify servers in an SSL transaction; SSL also allows for the use of **Client Certificates** so that clients can identify themselves when required.

By default, the traffic manager will not request a client certificate from clients that connect to a virtual server. If required, the virtual server can be configured to request that clients provide client certificates.

request_client_cert: You can specify whether the virtual server should request an identifying certificate from each client, and whether supplying a certificate should be optional or compulsory (i.e. all clients must present a valid client certificate). Note that if this is set to *not request* a certificate (the default setting), a client will be prevented from sending a certificate whether it wishes to or not.

client_cas: When client certificates are in use, the traffic manager must be able to verify that the certificate it is given is legitimate. It will inform the client which CAs it trusts and the client will send a certificate signed by one of these CAs (if available). Select the CAs as required.

If the client supplies a certificate which is not recognized by the CAs you specify, the connection is immediately closed and an SSL error is sent to the client.

ssl_client_cert_headers: Once the request has been decrypted, the traffic manager can optionally set HTTP headers that record the certificate fields from the SSL Certificate used, and if required additional certificate text from the SSL Certificate.

Header name	Sample Value
SSLClientCertVersion	3
SSLClientCertSerialNumber	ED41A8
SSLClientCertIssuer	C=US, ST=CA, L=San Jose, O=MyOrg
SSLClientCertSubject	C=US, ST=CA, L=San Jose, O=MyOrg
SSLClientCertNotValidBefore	1155304575 (unixtime)
SSLClientCertNotValidAfte	1186840575 (unixtime)

r	
SSLClientCertSubjectPublicKey	RSA (2048 bit)
SSLClientCertSignatureAlgorithm	md5withRSAEncryption
SSLClientCertHash	873FECCB50E0C1D34711ABE65BA70BA7

Finally, if you select the option to include the certificate text as well, the traffic manager will place the entire PEM encoded certificate in a header named `SSLClientCert`.

Configuring OCSP

The Online Certificate Status Protocol (OCSP) can be used to check the revocation status of certificates from a centralized server called an *OCSP responder*. It is commonly used as an alternative to Certificate Revocation Lists (CRLs). See the *Client Authentication* section of CHAPTER 13 for further details on OCSP.

The traffic manager provides a dedicated section under **Virtual Servers > SSL Decryption** for configuring OCSP. The following general settings are available:

ssl_use_ocsp	If set to yes, this virtual server will use OCSP to verify that client certificates have not been revoked.
ssl_ocsp_timeout	When contacting an OCSP responder, the traffic manager will wait for a response for the amount of time specified by this setting. If the OCSP responder has not replied within this time limit, the client's certificate will be rejected.
ssl_ocsp_max_response_age	If an OCSP response's 'thisUpdate' field is older than the number of seconds specified by this setting, the response will be considered invalid. If set to 0

	(zero), the 'nextUpdate' field of the response will be used to determine whether it has expired or not. If the time specified in the 'nextUpdate' field of the response has passed, the response is considered invalid regardless of this setting.
ssl_ocsp_time_tolerance	If the OCSP responder and traffic manager's clocks differ slightly then the times specified in the 'thisUpdate' and 'nextUpdate' fields of a response might cause the traffic manager to consider it invalid. If the responder's clock is a few seconds faster than the traffic manager's clock, for example, the 'thisUpdate' field of a response might appear to the traffic manager to be in the future. This setting allows you to specify the number of seconds to permit for clock differences.

The following OCSP responder settings are provided on a per-issuer basis. These allow you to specify different settings for certificates signed by particular issuers in the Certificate Authorities catalog.

Select a certificate authority in the **Add Issuer Specific Settings** section to add settings for that issuer. If the traffic manager receives a client certificate with an issuer that is not configured explicitly, the **Default Settings** will be used.

OCSP check	This setting is used to enable and disable OCSP checks for the issuer. Selecting 'Always' will cause remote connections to be dropped when an appropriate OCSP

	responder URL cannot be determined.
Use AIA for URL	Issuers can embed an OCSP responder URL into their client certificates using the X.509 *Authority Information Access* extension. If this setting is enabled, and the extension is available, the traffic manager will use it. Otherwise it will use the **Fallback URL**.
Fallback URL	If the responder URL cannot be determined by other means, this URL is used.
Request signing certificate	OCSP requests can be signed so the responder can identify and authenticate the source of the request. You can select a certificate uploaded to the SSL Certificates catalog to sign the request, choose to use the certificate configured under Default Settings, or disable request signing.
Use nonce extension	A unique sequence of data can be added to OCSP requests and responses to ensure an attacker cannot re-send a 'good' OCSP response after a client certificate is revoked. When using the nonce extension you can require the responder to always return a nonce by using 'strict' mode. Otherwise the traffic manager will accept the response if it does not contain the nonce. **Important:** Using this extension stops the responder from pre-generating responses to improve performance. Additionally, not all OCSP responders support this extension. If this is the case, a responder may return an

	'unauthorized' response.
Responder certificate	OCSP responses are signed by the responder to prove they are genuine. You can select a specific certificate that responses should be signed by, from those listed in the **SSL Certificate Authorities** catalog. You can instead configure the system to expect the response to be signed by the issuer's certificate by selecting "**Issuing Certificate**". Alternatively, you can select "**Signed by issuing certificate**". This means the certificate must either be the issuer's certificate itself, or be signed by it and have the `id-kp-OCSPSigning` marker in its **extendedKeyUsage** extension, and also have the **id-pkix-ocsp-nocheck** extension.

SSL Session ID cache

Once an SSL handshake has taken place, the SSL client (e.g. the web browser) and the SSL server (e.g. the traffic manager) record the encryption parameters using an SSL Session ID.

If the SSL TCP connection is terminated and the client reconnects, it may present its SSL Session ID. If the server recognizes the SSL Session ID, it can skip the computationally-expensive SSL handshake and resume the SSL session using the cached encryption parameters.

The traffic manager manages a cache of SSL session IDs. This cache is configured on the **System > Global Settings > SSL Configuration** section; you can configure the size of the cache (**ssl!cache!size**) and the expiry time (**ssl!cache!expiry**).

You can monitor the behavior of the SSL Session ID Cache using the **Activity > Current Activity** page. Select the **Cache values > SSL Session ID Cache** values to

plot on the current activity chart, or monitor these values using SNMP. The **Oldest** value will indicate the time since the least-recently-used entry in the cache was used. If the cache is full (comparing **Entries** with **EntriesMax**) and the **Oldest** value is significantly lower than the configured expiry time, then it is likely that your cache is too small and entries are being expired too early.

SSL Encryption

SSL encryption can be performed by a pool before it sends traffic to a back-end SSL server. When enabled, SSL encryption will use additional CPU resources in the traffic managers that perform the encryption.

To enable SSL encryption, edit the pool which you wish to encrypt traffic and go to the **SSL Settings** section.

The traffic manager now shows you the SSL settings for this pool.

ssl_encrypt: Enable SSL encryption to the back-end nodes by setting this to **Yes**.

ssl_strict_verify: As a protection against man-in-the-middle attacks and server spoofing, the traffic manager can validate the SSL certificates used by back-end servers against the certificate authorities in the catalog. If this is enabled, the traffic manager will reject connections to servers if their certificate has expired or if it has not been signed by a CA in the catalog.

ssl_client_auth: If the back-end server requires client certificate authentication, enable this setting to configure the traffic manager to select an appropriate certificate from its local Client Certificates catalog.

Click **Update** to complete the configuration and enable encryption to the back-end servers of this pool.

Preserving IP addresses with SSL forwarding

When the traffic manager forwards HTTP requests, it can optionally insert a header named `X-Cluster-Client-Ip` into the request so that downstream servers can determine the correct source IP address of the request. If the traffic manager decrypts an HTTPS request, it can also insert the `X-Cluster-Client-Ip` header into the request, even if it then re-encrypts the request.

However, if the traffic manager forwards an SSL request without decrypting and re-encrypting it, it cannot modify the data inside. This configuration is used with a

loopback virtual server, whereby the connections are load-balanced across a cluster of traffic manager systems for decryption. In this case, you may use the **ssl_enhance** setting in the pool to add a proprietary header to the SSL connection that contains key connection data that is not preserved, i.e. source IP and port and destination IP and port.

The traffic manager system that receives the 'enhanced' SSL connection must be configured with the **ssl_trust_magic** setting in the SSL decryption settings of the Virtual Servers. This setting will cause the traffic manager to strip out the proprietary header, and recognize the correct connection data – source IP and port, and destination IP and port.

The **ssl_enhance** setting can also be used when forwarding SSL connections to Zeus Web Servers, configured with the **ssl_trust_magic** setting.

Use of SSL Cryptographic Hardware

Stingray traffic management solutions support SSL hardware based on the RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki), such as nCipher's NetHSM and Sun's SCA 6000 card. The traffic manager also supports various Cavium PCI cards (CN 1000 and CN 2000 series) with a dedicated driver.

This hardware offloads SSL computation (the RSA private key decryption) from the traffic manager system's CPU onto the SSL cryptographic hardware. Some PKCS#11 devices also provide hardware key management, so that the private key is stored securely on the hardware device and cannot be accessed directly without the correct authentication.

Although the majority of SSL hardware devices require explicit configuration as described below, the traffic manager automatically detects and uses the SSL hardware support present in UltraSPARC T2 processors.

Note: Some variants of Stingray, such as the virtual appliance, may display a limited set of the following options as the hardware is not supported for these versions.

Note: Use of SSL Hardware may not improve the overall performance of your traffic manager system. Although they offload the RSA calculation from the main CPU cores, the overhead in communicating with the hardware device is not negligible. The traffic manager is able to perform many thousands or tens-of-thousands of SSL calculations on general purposes CPUs. The primary benefit of many SSL hardware devices is their ability to store the private key securely.

Configuring the traffic manager to use an SSL hardware device

Go to **System> Global settings** and click on **SSL Hardware Support**. To configure the traffic manager to use SSL hardware, you must first select the SSL library.

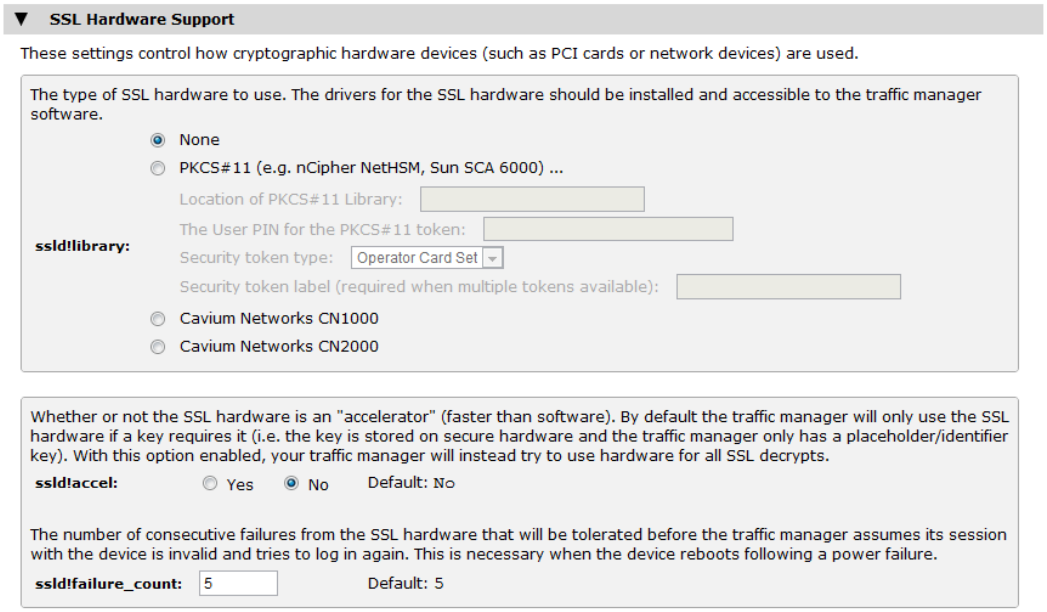


Fig. 32. Settings for using SSL hardware accelerator

The following describes the keys common to all library types:

`ssld!library`	Set to PKCS#11 for generic PKCS compliant hardware such as nCipher NetHSM or Sun SCA 6000, or set to the appropriate Cavium Driver to use a supported CN 1000 or CN 2000 series Cavium device. Additional settings for PKCS#11 are: **Location:** the traffic manager will search standard system locations for the PKCS#11 interface library; if the library is located elsewhere, you can specify the location here. **User PIN:** the PIN required to create a user session on the SSL hardware.
`ssld!accel`	Enable this setting to use always the SSL hardware to perform RSA decryptions. If this setting is disabled, the traffic manager will only use the SSL hardware when the private key is stored securely

	on it.
`ssld!failure_count`	Enter the consecutive number of attempts that the traffic manager will run before assuming the session is invalid and tries to log in again (due to reboot, power failure, etc.).

Verifying correct operation of SSL hardware

The **Diagnose** page will check that the traffic manager can communicate with the SSL hardware. Any errors will be indicated on that page.

If you have configured the traffic manager to use a hardware acceleration device that supports hardware key management, you can then:

Create new private keys (and corresponding public certificates) on that hardware using the traffic manager. Follow the procedure referred to in the *Creating a new self-signed SSL Certificate* section of CHAPTER 13 to create a new self-signed Server or Client SSL certificate, making sure to select the "Create private key on hardware" option that is present when a key-management device is configured;

Manually install private keys and public certificates from the hardware on the traffic manager. Follow the instructions for your hardware device to extract an encoded version of the private key and the public certificate; then install them on the traffic manager using the **Import Certificate** tool in the relevant **Server/Client SSL Certificates** catalog.

> The traffic manager will verify that it can access and use any hardware-managed keys, and will indicate an error on the **System > Diagnose** page and the **Event log** if any problems occur.

Note: Although Client SSL Certificates and corresponding private keys may be marked as *unavailable* on the **Diagnose page**/**Event Log**, the pool using them cannot be marked as having any problems because the traffic manager never knows what client certificate will be asked for until it connects to a back-end SSL node that requires one.

CHAPTER 14 Health Monitoring

This chapter describes the heath monitoring capabilities found within Stingray, which can be used to monitor server nodes for correct operation and raise alerts and route round failed nodes when an error is detected.

Which nodes are monitored?

The traffic manager uses two methods to monitor the correct operation of nodes: passive monitoring (checking the status of a node when it is used) and health monitoring (additional tests that are run against a node on a periodic basis).

Passive monitoring tests are only performed against nodes that are actively in use.

Health monitors are executed against all of the nodes in a pool. They are only run against pools that are in use:

- Pools that are configured as the default pool for a virtual server;
- Pools that are explicitly referenced by name in a TrafficScript or RuleBuilder rule, either by the `pool.select()` or `pool.use()` functions;
- Pools that are configured as the failpool for a pool that is in use.

If a pool is not referenced by the traffic manager configuration in this way, it is considered to be 'not in use' and is not monitored using health monitors. Any failures of nodes will not be detected.

Note: Pools must be explicitly referenced by name when they are used in a TrafficScript rule. By default, referencing by a variable (`pool.use( $name );`) or any other means is not permitted by the traffic manager.

If you enable the setting **trafficscript!variable_pool_use** (in the **Global Settings** page), you may use variables for pool names. If this setting is enabled, the traffic manager will execute health monitors against all of the pools you have configured, not just the ones that are clearly in use.

Using nodes in multiple pools

The same node (IP address and port) may be referenced in several different pools. If a node has failed in one pool, it is not used by the load balancing or session

persistence decisions. However, that node may be used in other pools until the passive monitors and health monitors assigned to those pools report a failure.

Example

Your web application has 4 back-end servers (nodes). Each node hosts the same dynamic content and static content. If the dynamic content on a node fails (for example, the java servlet crashes), you may still want to use that node for other static content:

Create two pools named 'Dynamic' and 'Static', each containing all 4 nodes.

Create a rule that uses pool 'Dynamic' for dynamic content (Java Servlets, PHP and ASP files etc) and uses pool 'Static' for all other content.

If a node fails in the 'Dynamic' pool and fails to send a valid HTTP response, the passive monitoring used by that pool will determine that node has failed. No more traffic from the 'Dynamic' pool will be sent to that node.

However, the 'Static' pool will continue to send traffic to that node, as long as it returns valid HTTP responses for requests from the 'Static' pool.

For more fine-grained detection of errors, you can assign different health monitors to each pool. For example, the health monitors for the 'Dynamic' pool can send synthetic PHP or ASP requests; if these monitors fail, the node will considered to have failed in the 'Dynamic' pool, but not in other pools.

Passive Health Monitoring

A pool performs a set of checks every time it attempts to send a request to a node; this process is referred to as 'Passive Monitoring':

- The traffic manager attempts to connect to a node; if the connection is refused, or is not established within the **max_connect_time** setting (default 4 seconds), the request is considered to have failed;
- The traffic manager writes the request data down the connection; if the connection is closed prematurely, or if the beginning of a response is not received within **max_reply_time** seconds (default 30 seconds), the request is considered to have failed;
- SSL only: if the SSL handshake to the node fails, the request is considered to have failed;

- HTTP only; if an invalid HTTP response is received, the request is considered to have failed.

Note: The **max_connect_time** and **max_reply_time** settings are properties of the **Connection Management** settings (see the *Connection Management* section of CHAPTER 5) in a Pool.

Retrying failed requests

If these checks fail, the pool *may* try the request against a different working node, and *may* try every node in the pool before abandoning the request. The behavior is determined by the *Idempotent* status of the request.

By default requests are assumed to be idempotent, i.e., they can be safely retried multiple times without undesired side effects. An exception to this is any request received through a virtual server using one of the *generic*-type protocols ("Generic Client First", "Generic Server First", "Generic Streaming"). In order to be idempotent by default, an end-point to the request must first be defined[9] in order for failure to be measured and retries to be triggered.

RFC 2616 defines some HTTP requests as non-idempotent[10] (e.g., they may cause a transaction to take place or change state on the server). The traffic manager follows these recommendations and will treat HTTP GET, HEAD, PUT, DELETE, OPTIONS and TRACE methods as idempotent; all other requests are considered non-idempotent.

- **Idempotent (no side effects)**: the traffic manager will retry these requests against other believed-to-be working nodes, and may try every node in the pool before abandoning the request;
- **Non-idempotent (side effects)**: the traffic manager will only retry a non-idempotent request if it failed to open a TCP connection to the failed node.

Note: When the traffic manager establishes a TCP connection, it immediately writes the request data down that connection. The traffic manager is not able to determine whether or not the node has received the request data and begun processing it. Therefore, non-idempotent requests are only retried if the connection could not be established in the first place.

[9] Achieved via TrafficScript, using functions such as `request.endsWith()` or `request.endsAt()`

[10] http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html

You can override the idempotent/non-idempotent decision made by the traffic manager by using the `request.setIdempotent()` and `http.setIdempotent()` TrafficScript functions to indicate to the traffic manager that a particular request should be considered safe to retry.

503 Server Errors

503 Server Error responses are treated differently, because you typically would not wish to try a request that generated a server error against every node in your cluster.

A request that generates a 503 Server Error is only retried if:

- Passive monitoring is enabled
- No session persistence was in place
- The request is idempotent (no side effects)

A request is tried a maximum of three times before the 503 error is sent back to the client.

Node failures

The traffic manager will infer that a node has failed if connections to that node fail consistently, with **node_connection_attempts** (default 3) failures in a row with no intermediate successful transactions. Once a node has been deemed to have failed, it is not used for at least **node_fail_time** seconds (default 60 seconds), after which it is tentatively used to determine if it has recovered.

Note: The node_connection_attempts and node_fail_time settings are found on the Pool > Connection Management page.

You may wish to try enabling the **log!server_connection_failures** setting in the **Connection Error Settings** section on the **Virtual Server > Connection Management** page. This could help provide useful log information regarding what the actual node failures were. See the *Handling Errors* section of CHAPTER 4 for more details.

Enabling and Disabling Passive Monitoring

Passive Monitoring is used by default, but can be disabled on a per-Pool basis using the **passive_monitoring** setting in the **Monitors** section of a Pool configuration. If this setting is disabled, you should ensure there are suitable **Health Monitors** configured otherwise failed requests will not be detected and subsequently retried.

Overview of Health Monitors

In addition to the inferences from passive monitoring, a pool may be configured with explicit **Health Monitors** that perform periodic tests and verify the correct operation of each node. By using health monitors, the traffic manager can detect failures even when no traffic is being handled.

The tests are performed at configured intervals against each node in a pool; if they fail sufficiently often, that node is marked as unavailable, and the traffic manager will stop sending traffic to that node. Pool-wide monitors just test a single machine and can be used to indicate that an entire pool is down.

The monitors are held in the Monitors Catalog and can be applied to any pool. Each performs a specific test. These range from simple tests, such as pinging each node, to more sophisticated tests which check that the appropriate port is open and the node is able to serve specified data, such as your home page.

The Monitors Catalog

Setting up a Monitor

To access the Monitors Catalog, click the **Catalogs** button on any page of the Admin Server interface. You can unfold the **Monitors Catalog** section to see the monitors you have already set up. Clicking on **Monitors Catalog** takes you into the catalog.

The traffic manager contains a number of preset monitors. Some of these deal with generic TCP client-first or server-first protocols, and others are protocol-specific. For instance, the built-in **Full HTTP** monitor requests a specified page from a web server and looks for a suitable status code in the response.

All monitors have the following configurable settings:

delay	Monitors are executed repeatedly against each node. Once a monitor has completed or times out, the traffic manager pauses for this configured period of time before trying the monitor again.
timeout	If a monitor fails to complete within this timeout period, it is aborted and judged to have failed.
failures	A monitor may occasionally fail for unpredictable and unrepeatable reasons, such as a brief network glitch. The failures setting indicates how may times a monitor must fail consistently before the monitored node is judged to have failed.

verbose	This setting enables verbose monitor logging in the error log. It is useful for testing a monitor, but should not be used in a product system or for a long period of time because it cal fill up the error log file.

You can edit the built-in monitors: click on the name of one, and modify the timings, failure conditions, and logging, along with any other monitor-specific settings (such as the web page to request). Click **Update** to commit your changes.

You can also click **Copy Monitor** to copy the built-in monitor under a new name, and make your changes to the copy.

Built-in Health Monitors

The traffic manager contains a number of predefined monitors in the **Monitors Catalog**. These 'built-in' monitors perform standard tests for various protocols.

Basic Monitors

Basic Monitors perform simple tests against a node.

Ping: The ping monitor performs ping requests against each node. It will fail if no response is received.

Connect: This performs TCP connects to the node. Its purpose is to ensure that a server is listening on the port, and will accept traffic. This monitor can cause some programs (such as exim) to output errors; in this case a more protocol specific monitor should be used.

Advanced Monitors

Advanced Monitors perform more complex request/response tests against a node. These monitors are used as building blocks for other monitors, and can use SSL to connect and transfer the request and response.

Client First: This is a monitor based on the TCP transaction monitor. It will send some data to the node, and match the response against the supplied regular expression. In order to use it, the data to send (the **write_string**) must be configured.

Server First: This monitor can be used to check services where the server writes data first. It will connect to the node, and check that the server returns some data. This monitor can easily be altered to check for specific data being returned by the server.

Full HTTP / HTTPS: This monitor can be used to perform HTTP or HTTPS requests. The host header, URL and authorization parameters can all be configured.

The response return code is matched against the **status_regex**: if you are expecting a 200 return code, the **status_regex** should be "^200$". For instance, ^[234][0-9][0-9]$ will accept errors from 200 to 499.

The response body is matched against the optional **body_regex**.

If the pool that an HTTPS monitor is assigned to has the **ssl_strict_verify** option enabled (to check the validity of the node's certificate), then the monitor will detect this and perform the certificate verification.

Protocol-specific Monitors

Protocol-specific monitors perform specific tests against a node.

- **DNS:** This is an example of a program monitor. It will run a program (`ZEUSHOME/monitors/dns.pl`) that makes a DNS request to the node, and validate the address returned.
- **FTP:** This monitor is based on the TCP transaction monitor. It will make a simple FTP connection, and check that the server is responding correctly. It only checks that the banner is returned by the server, and does not authenticate.
- **POP:** This monitor is based on the TCP transaction monitor. It will make an initial connection to the node, and check that the correct POP banner is returned.
- **SMTP:** This is identical to the POP monitor, except that it checks for the SMTP server banner being returned.
- **Simple HTTP and HTTPS:** These monitors request the home page from each node. They check that the response code is a 2xx, 3xx, or 4xx response code, and indicate a failure of not.
- **RTSP:** this monitor will send an RTSP DESCRIBE request to the server. It will check for a 2xx, 3xx or 4xx response or report a failure if one is not received. Note that some RTSP servers require the presence of a path in such a DESCRIBE request, in which case you should enter one

in the rtsp_path setting. The protocol specific additional settings for RTSP are:

rtsp_path: the path to be used in the DESCRIBE request. This is the path to the multimedia file being streamed.

rtsp_status_regex: the status code regular expression that the RTSP response must match.

rtsp_body_regex: the regular expression that the RTSP response body must match.

- **SIP (UDP):** this monitor will send a SIP OPTIONS request to the server with

  Max-Forwards set to 0. It will check for a 2xx, 3xx or 4xx response and report a failure if one is not received.

  sip_status_regex: the status code regular expression that the response must match.

  sip_body_regex: the regular expression that the SIP response body must match.

  udp_accept_all: should this monitor accept responses from any IP and port?

- **SIP (TCP):** this monitor uses the same request as the SIP-UDP monitor but sends it over a TCP connection.

Custom Health Monitors

The main **Monitors Catalog** page has a **Create New Monitor** section, where you can create an entirely new monitor. There are several options for the underlying type:

TCP Transaction	This performs a TCP transaction with the target machine, with an optional string of data to write to the connection It can look for a specified regex in the response.
Ping	This pings the target machine at specified intervals.

External Program	This runs an external program, whose file path on the traffic manager machine you specify. It sends two built-in arguments, `port` and `ipaddr`, along with any others specified. An exit code of 0 is classed as a success; a code other than 0, or a timeout, is a failure. *ZEUSHOME*`/zxtm/conf/scripts` contains your custom monitor executables; you can see an example there.
HTTP	This sends an HTTP request to the target server, optionally using SSL, with specified parameters such as host header and the URL path to use. It searches for a status code regex in the response code, and a regex against the response body data.
TCP Connect	This makes a TCP connection with the target machine, to check that the appropriate port is open.
RTSP	This issues an RTSP DESCRIBE request to the target server, with the specified **rtsp_path** parameter. It looks for a status regex in the response, and a regex against the response body data.
SIP	This sends a SIP OPTIONS request to the target server using the specified transport protocol. It searches for the status regex in the response code and the body regex in any body data that was received. A failure is reported if either of these do not match.

When you have chosen the basic settings for your monitor, click **Add Monitor**. You can click on the name of the monitor to edit it.

It is possible to build sophisticated custom monitors, using either the **TCP Transaction** or **HTTP** monitor templates. For example, a pair of HTTP-based monitors could test the secure part of a website to verify that:

a) A request with a valid username and password receives a valid 200 response;

b) A request with an incorrect username and password receives a 401 or 403 error response.

TCP Transaction monitors are useful building blocks for nodes which use other protocols.

For more sophisticated tests, a monitor can run an external program or script. **External Program monitors** are described in the *External Program Monitors* section of CHAPTER 14.

Per-node and Pool-wide Monitors

Monitors fall into two categories: ***per-node*** and ***pool-wide***. A per-node monitor tests the health of each node in the pool. A pool-wide monitor performs tests on one machine which influences the health of the entire pool. For example, a mail server pool might keep its data on an NFS server, which each of your back-end servers accesses. A pool-wide monitor could test this server. If it fails, none of the back-ends can retrieve the data so the whole pool is deemed to have failed.

You can choose whether your monitor should be per-node or pool-wide. If it is pool-wide, you must specify a machine (and possibly port) for the monitor to test, such as an NFS server used by the pool.

Using Health Monitors

Applying a Monitor to a Pool

To apply a monitor to a pool, click the **Services** button and then the **Pools** tab. Click on the name of a pool to edit it.

Click Health Monitoring to choose the monitors used by the pool. You can add a new monitor from the drop-down list, clicking **Add Monitor** when you have finished. To change settings for a monitor attached to your pool, click the **Edit In Catalog** link against it.

Note: A monitor sending test requests will increase the load on a pool, albeit by a small amount. You should ensure that your back-end servers can handle the traffic produced by the monitors, to avoid a DoS failure.

External Program Monitors

An external program monitor can be written in any language. The traffic manager passes command-line arguments to the executable:

```
     domonitor --ipaddr=<machine_to_monitor>  --
node=<nodename> \
               --port=<port_to_monitor>
```

Note: The `ipaddr`, `node` and `port` arguments are always passed to an external program monitor. The `node` argument is the hostname part of the node, as configured in the pool, and the `ipaddr` is the corresponding IP address.

You can also specify additional arguments for the monitor, which may be useful if you are using the same executable for several different monitoring tasks. Use the **Program Arguments** section of the **Edit** page for the monitor in question.

For example, if you configured two additional arguments (named 'regex' and 'interface') and gave then appropriate values on the Edit page, the monitor executable would be invoked as follows:

```
     domonitor --ipaddr=<machine_to_monitor>  --
node=<nodename> \
               --port=<port_to_monitor> \
               --interface=<interface_value> --
regex=<regex_value>
```

A successful monitor should exit with an exit code of 0; a timeout or non-zero exit code is interpreted as failure, and in this case the contents of STDERR are written to the Event Log.

When you develop and test a monitor executable or script, you can easily run it from the command line, providing arguments in the format described above. Your monitor can emit debugging information to STDOUT.

When the monitor is used by the traffic manager, you can enable verbose mode. If the monitor is called in verbose mode, anything printed to STDOUT by the executable is written to the Event Log; verbose mode can be toggled on the monitor's Edit page.

Uploading Monitors to the traffic manager

You should use the **Catalog > Extra Files > Monitor Programs** page in the Stingray Administration Interface to upload, manage and delete monitors on the traffic manager:

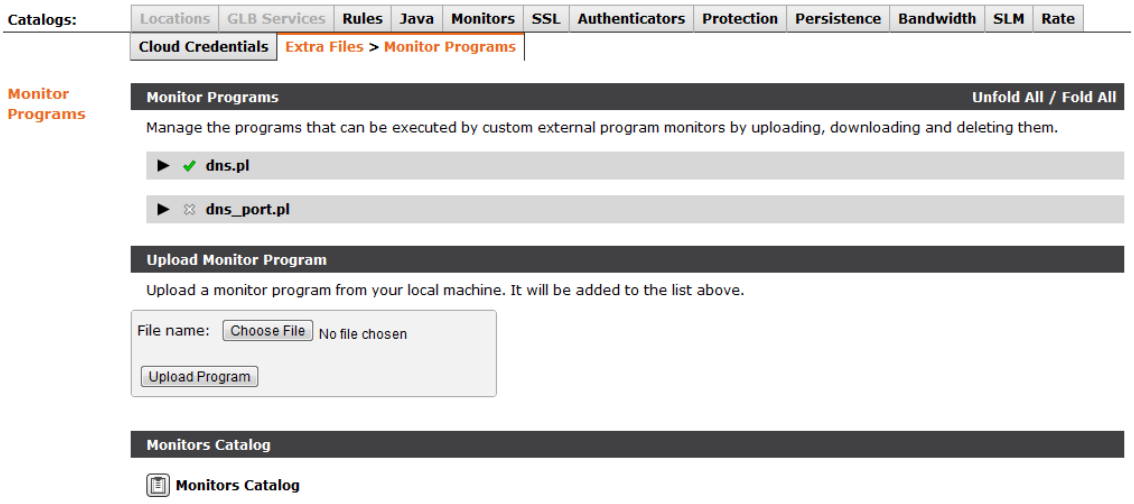


Fig. 33. Managing External Monitor programs

Writing Monitors in Perl

If you choose to write your monitor in Perl, Stingray provides a helper module, `Monitor.pm`, that is included with your traffic manager. This includes a `ParseArguments()` function, as well as functions to exit with success or failure and to write out logs.

```
use Zeus::ZXTM::Monitor qw( ParseArguments
                            MonitorWorked
                            MonitorFailed
                            Log );

%ARGS = ParseArguments();

Log( "Running test" );
if( do_test( $ARGS{ipaddr}, $ARGS{port} ) )
{
    MonitorWorked();
} else {
    MonitorFailed( $error_message );
}
```

For more information on `Monitor.pm`, use this command:

```
$ perldoc ZEUSHOME/zxtm/lib/perl/Zeus/ZXTM/Monitor.pm
```

CHAPTER 15 Service Protection

This chapter explains some of the risks associated with Internet hosting, and how to use the traffic manager to mitigate those risks to your services.

Classes of Risk

Denial of Service (DoS)

A ***Denial of Service*** or ***DoS*** attack is characterized by a malicious attempt to prevent legitimate use of a service. This could take the form of attempting to flood a network, or disrupt connectivity between machines. It may be designed to consume the resources normally available for specific users or applications, for example exhausting the available CPU, bandwidth or storage of a server or cluster of servers.

DoS attacks are based around exploiting design weaknesses, flaws in the operating system, or services with unbounded access to a resource such as CPU time, disk or memory. For example, a malicious attacker may find a way to craft a request to a database-driven website that requires extremely intensive SQL operations to complete. This could effectively deny access to the database for the duration of the query.

DoS attacks that are delivered through exposed services such as websites are the most common class of risk. The traffic manager is designed to assist in mitigating the effects of such attacks, or preventing them altogether.

Web Worms and Viruses

Vulnerabilities in operating systems have lead to many Internet ***worms*** which propagate by installing themselves, using the weakness as a point of entry. For example, a worm can install a payload sufficient to propagate itself further. Worms also affect desktop PCs. The frequency of this class of attack is increasing, and is often part of a planned DDoS attack in which the compromised machine will play a part.

In addition to worms, users of desktop PCs are exposed to risks from viruses, often received by email or from unscrupulous websites. These viruses may install services on the PCs which can be controlled remotely.

Distributed Denial of Service Attacks (DDoS)

Distributed Denial of Service attacks use large numbers of client machines, often "recruited" after being compromised by worms or viruses. A malicious attacker can

control these clients remotely, making the combined impact of a focused attack on an individual service provider or business far greater than a conventional DoS attack. Mitigating distributed attacks is far more challenging than handling a DoS attack, as it requires the cooperation (e.g. outbound packet filtering) of other ISPs to fully address the issue.

The traffic manager can provide protection against basic DDoS attacks as well as DoS attacks, provided network bandwidth is not flooded and provided there is sufficient capacity to manage the incoming connections.

Malformed HTTP Attacks

Even correctly firewall-protected web servers must still present their public services on port 80 to the Internet. HTTP's client-server architecture means that clients can send data to these web servers freely through port 80. This information can be crafted to maliciously overload or subvert a server. Attacks exist where the request is malformed in such a way as to exploit bugs and compromise the web server to give control to a remote attacker.

Firewalls and Other Security Measures

Note that the traffic manager is not a firewall; it is intended to be used in conjunction with a dedicated firewall. See CHAPTER 23 for a full discussion of secure operation.

Protection Features

Stingray traffic management solutions are robust products that are not affected by known classes of exploits. For example, they are believed to be invulnerable to all known worms and malformed HTTP attacks. In addition, since the traffic manager is placed at the point of ingress for traffic to your clusters - and is able to make intelligent routing decisions - it can help defend your platform from obvious network attacks and filter malicious content transparently. This is particularly effective if your back-end servers use a private network rather than having Internet IPs.

Stingray is also equipped with ***Service Protection*** features, which can be configured for each virtual server independently. These features allow requests to be managed and filtered dynamically in a number of ways.

Network Access Restrictions

The source IP address of a client can be used to decide whether or not to accept requests. If a particular IP address or network block is generating malicious requests,

the traffic manager can be configured to drop all connections efficiently from these addresses, thus protecting the back-end pools from attack.

Connection Limiting

Often, clients abusing a system generate abnormally large numbers of connections. The traffic manager is able to detect and filter unusually heavy activity, based on the traffic from the top 10 busiest IP addresses or on the number of connections per minute being made from each IP address. This allows the traffic manager to deny access to clients making overly intensive requests to your systems.

Malformed HTTP Filtering

The traffic manager is able to detect and reject certain classes of malformed HTTP requests, and enforce standards-compliant requests from clients. For instance, the traffic manager can filter binary data from requests and prevent very large headers being used as a vector for a DoS attack.

Rule-Based Protection

The Stingray service protection system can be configured to screen all incoming requests so that any matching the specified criteria are dropped. This functionality can be used to protect your system against known vulnerabilities in third-party applications running on your websites. For example, some business-critical applications such as shopping carts are sometimes found to have insecurities that are triggered by requesting a URL containing badly-formed parameters.

The forms of these requests are often made public on security mailing lists. The Stingray service protection system enables you to use this information to filter out any requests containing these badly-formed parameters, so that you can resolve these vulnerabilities quickly and easily across all your websites.

You can configure the traffic manager to block requests by building up a list of rules that are used to filter all incoming requests. In addition you can limit the sizes of requests and the format of the information they can contain.

Enabling Service Protection

Service Protection is configured in two stages. Firstly, ***service protection classes*** are created in the **Catalog**. You can configure as many service protection classes as you require, each with its own settings and rules.

Secondly, the service protection class is assigned to one or more virtual servers, thus associating your protection settings with requests for that virtual server. You can assign the same service protection class to many different virtual servers.

Adding a Service Protection Class

In order to add a service protection class, first click on the **Catalogs** button. Select **Service Protection catalog**.

The traffic manager now shows you a list of existing service protection classes, and allows you to add a new service protection class by entering a name and clicking **Create Class**.

When you add a new service protection class, there are five main configuration groups.

Basic Settings

These settings enable and disable service protection and dictate in which mode the service protection class operates.

If you wish to bypass this service protection class temporarily, you can enable or disable service protection for the class. This is particularly useful if many virtual servers are configured to use the service protection class.

You can configure the log buffer time, which is the amount of time the traffic manager buffers logging in order to reduce disk writes (and reduce any knock-on problems if heavy logging results during a DoS attack). The traffic manager also includes a debug option, which causes verbose output to be logged.

Important: The debug option can create heavy loads on your traffic managers, and should not be used in production systems.

The traffic manager also offers a test mode, whereby the decisions made by the traffic manager with this service protection class are logged, but not acted upon. This option is designed to help tune the parameters of the service protection class to suit your services, and should be used for a period of testing before switching on protection.

Connection Limiting Settings

These settings specify limits on the number of connections that the traffic manager will allow from individual IP addresses.

You first supply a minimum and maximum number of connections, which the traffic manager will always allow. The minimum (`min_connections`) is the number of simultaneous connections which the traffic manager will always allow from one IP, even when limiting the connections from the top 10 IP addresses (see below). The maximum (`max_1_connections`) is the maximum number of connections which will ever be allowed from one IP.

You can then specify the maximum number of connections from the top 10 IP addresses. The top 10 clients will normally represent the majority of the load on a particular virtual server if any form of DoS attack is in progress or other abusive connections are being made. It is therefore desirable to limit this particular group of users while leaving other requests untouched.

Lastly, the total rate of connections (`max_connection_rate`) that any individual IP address may make can be limited (or set to zero for unlimited). You can configure the interval that this rate is assessed – connections per second, or connections per minute. Any connections that exceed this rate are immediately dropped.

Note: In the case of HTTP, each individual request is counted as a connection, even if they are tunnelled within a single Keepalive connection.

Each traffic manager maintains its own table of IP addresses. Note that the IP address may not uniquely identify a particular client, for example if the client is behind a proxy.

For more fine-grained connection rate control, you can use the Request Rate Shaping feature described in CHAPTER 17.

Access Restrictions

The **Allowed** list allows you to specify allowed IPs or networks which are exempt from the normal service protection checks in this service protection class. **Banned** IPs are IPs or networks from which connections are always dropped.

To add to either of these lists, input the IP or network address in the appropriate box. Entries can be in one of the following formats:

```
10.0.1.0/255.255.255.0
10.0.1.0/24
10.0.1.1
```

HTTP-Specific Settings

These settings provide extra checks which can be performed on HTTP traffic.

The traffic manager allows you to enforce compliance with RFC2396; this checks that the URL is properly formed according to accepted guidelines. Since standard web browsers comply with RFC2396, this test will reject requests that include unusual or malicious data or formatting, while allowing legitimate requests to pass through.

The length of the HTTP body and any HTTP header can be restricted. This prevents a malicious attacker sending a very large or endless body or header. Web servers normally buffer incoming requests in progress in memory, and can therefore be vulnerable to a memory depletion attack. You can guard against this by preventing clients from sending large amounts of request data.

You can also specify a limit to the total size of all the HTTP headers. This prevents an attacker from constructing a very large request using a large number of HTTP headers.

You can limit the size of the URL requested, to prevent buffer overflow attacks and other attempts to use the URL as a vector for large amounts of data or worms.

After decoding, the traffic manager can reject HTTP headers that contain binary data. This is often an indicator of an attempted buffer-overrun attack, although some poorly-written applications use binary data. If you suspect that your applications could be affected, you should first prove this feature using the test mode.

Lastly, the traffic manager allows you to specify whether to reject requests silently if they fail one of the above tests, or to send HTTP error messages to the client that makes the request. Sending error messages incurs slightly higher overheads, which can be significant if a DoS attack is in progress.

Note: If you associate this service protection class with a virtual server that does not handle HTTP traffic, then these settings will be ignored.

Service Protection Rule

This section displays the current rule, if any, which the service protection class is using. If you wish to change the rule, select the rule you wish to use from the drop-down menu. For more information on TrafficScript, please see CHAPTER 8.

To commit your settings and create the service protection class, click **Update**.

Applying a Service Protection Class to a Virtual Server

In order for a service protection class to take effect, it must first be assigned to a virtual server. To do this, click on the **Services** button. Next, click on the **Virtual Servers** tab and choose an existing virtual server.

Select **Service Protection**. The user interface offers a drop-down box of the service protection classes that are available in the catalog.

Once you have selected a service protection class, click **Update.**

The traffic manager applies the new service protection class immediately across all traffic managers. A summary table indicating whether the service protection class is in test or debug mode is shown. If the service protection class has not been enabled (see the *Basic Settings* section of CHAPTER 15) then this is also indicated.

Service Protection Performance

Stingray's service protection system requires additional memory and CPU time for each request. If you are managing a large number of requests and use sophisticated rules in your service protection class, every request for a virtual server using this service protection class will take slightly longer. For large systems it may be necessary to add further traffic managers to your front-end network in order to provide additional capacity.

CHAPTER 16 Bandwidth Management

This chapter explains what Bandwidth Management is, and how to configure your traffic manager to control how bandwidth is used on your network.

Note: Bandwidth Management is not available on all Stingray traffic management solutions. If required, it can be obtained via a software or license key upgrade.

What is Bandwidth Management?

Bandwidth management allows the traffic manager to limit the number of bytes per second used by inbound or outbound traffic, for a Virtual Server or by the type of request.

Normally, network bandwidth is provided at the highest rate possible for all connections. This may result in uneven use of your network, possibly with too much bandwidth being used by secondary services at the expense of your most critical services. Bandwidth management with the traffic manager allows you to control this imbalance explicitly.

For example, you may have a 20Mbits/s network connection which is being over-utilised for FTP downloads, which is affecting the responsiveness of the main HTTP service. You may therefore wish to limit the bandwidth to the FTP virtual server to 2Mbits/s.

Bandwidth limits are automatically shared and enforced across all the traffic manager machines in a cluster. Individual traffic manager machines take different proportions of the total limit, depending on the load on each, and unused bandwidth is equitably allocated across the cluster depending on the need of each machine.

Note: Bandwidth management is only applicable to traffic sent by the traffic manager. In other words, the traffic manager can control the bandwidth used when writing requests to server nodes, and when writing responses back to clients. The traffic manager CANNOT control how quickly clients write the requests to the traffic manager system, or how quickly servers attempt to write responses to the traffic manager.

Bandwidth management only works on data sent to external addresses. It does not apply to data transferred internally within the traffic manager, such as with configurations where the data is passed from one pool to another internal virtual server. In these cases, the host system will use the system's loopback network interface which is not subject to bandwidth restrictions.

Configuring Bandwidth Management

The traffic manager's bandwidth management features are provided via the Catalog. This means that you may add more than one Bandwidth Management Class, each with its own limits. It also means that the choice of Bandwidth Management Class can be made through a TrafficScript rule, providing greater flexibility for situations where it is helpful to distinguish between requests to the same Virtual Server. For example, the traffic manager's Bandwidth Management system can be used to apply different limits for CGI requests than for image requests (see examples, below).

If you have a cluster of more than one Traffic Manager, the Bandwidth Management system takes this into account. You do not need to make adjustments when new servers are being added, as this is done automatically.

Adding a Bandwidth Class to the Catalog

To add a Bandwidth Management Class to the Catalog, click on **Catalogs** and then click **Bandwidth** in the menu bar.

The traffic manager now shows a list of the existing Bandwidth Management Classes, if any, and provides a box where you can create a new Bandwidth Management Class.

Type a name into the box and click **Create Class**. The traffic manager now displays the settings page for this new class. You can select the maximum bandwidth and the limit sharing options.

To select the maximum bandwidth limit you can either:

Select a predefined value from the drop-down list.

Select Custom from the drop-down list and then enter your value, in kbits/second, in the box provided.

There are three possible modes for how the bandwidth class shares out bandwidth between connections:

Each connection is limited to the maximum rate: No sharing is enforced, and all users of this class on all traffic managers will adhere to the maximum bandwidth limit set. This is useful when delivering video / audio streams to individual clients.

Bandwidth is shared per traffic manager: The limit is shared amongst all users of the bandwidth class on each separate traffic manager. If you have a limit of 1mbps and 3 traffic managers, each could deliver up to 1mbps, giving a total of 3mbps.

Bandwidth is shared across all traffic managers: The limit is shared amongst all users of the bandwidth class across all traffic managers. With a limit of 1mbps and 3 traffic managers, the cluster will only deliver 1mbps total. Note that the limit is allocated dynamically - each traffic manager will be capable of using the full 1mbps if the others are idle.

Click **Update** to commit your changes.

Note: Unlike storage or memory capacity, network throughput is conventionally measured in factors of 1000 (not 1024).

Assigning a Bandwidth Class to a Virtual Server

A bandwidth management class that is assigned to a *virtual server* will limit the bandwidth that the virtual server can use when sending response data to a client.

Configuration

To assign a Bandwidth Management Class to a Virtual Server, click on **Services**, then **Virtual Servers**. Choose the Virtual Server from the list of available servers, and click on the **Edit** button.

Next to **Assigned Classes**, click **Edit**. Now click on **Bandwidth Management**, and use the radio buttons to select one of the Bandwidth Management Classes.

Click **Update** at the bottom of the page to commit your change. The Virtual Server will begin using this Bandwidth Management Class immediately.

Note: if you do not have any Bandwidth Management Classes, the only option displayed will be ***none***. You can add a Bandwidth Management Class by clicking the **Create New Bandwidth Management Class** link.

Assigning a Bandwidth Class to a Pool

A bandwidth management class that is assigned to a *pool* will limit the bandwidth that the pool can use when sending request data to a node.

Configuration

To assign a Bandwidth Management Class to a Pool, click on **Services**, then **Pools**. Choose the Pool from the list of available servers, and click on the **Edit** button.

In the **Bandwidth Management** section, click **Edit**. Use the radio buttons to select one of the Bandwidth Management Classes.

Click **Update** at the bottom of the page to commit your change. The Pool will begin using this Bandwidth Management Class immediately.

Note: if you do not have any Bandwidth Management Classes, the only option displayed will be ***none***. You can add a Bandwidth Management Class by clicking the **Create New Bandwidth Management Class** link.

Using TrafficScript to select a Bandwidth Class

The bandwidth management class can also be selected per request, using either of the TrafficScript functions `request.setBandwidthClass()` and `response.setBandwidthClass()`. These functions affect the 'to-node' and 'to-client' bandwidth respectively.

A Bandwidth Management Class can be applied at any point in a TrafficScript rule, and will override the Virtual Server's own settings if they exist.

For example, the following TrafficScript response rule will choose the 'downloads' Bandwidth Management Class for responses which are large, or are from the "/downloads/" part of the site:

```
# This must be used as a response rule,
because we

# want to determine the response content
length...

$url = http.getPath();

$clen = http.getResponseHeader( "Content-
Length" );


if( string.startsWith( $url, "/downloads")
||

      $clen >= 100*1024 ) {

   response.setBandwidthClass( "downloads"
);

}
```

CHAPTER 17 Request Rate Shaping

This chapter explains what Request Rate Shaping is, and how to configure the traffic manager to rate-limit requests to your applications.

Note: Request Rate Shaping is not available on all Stingray traffic management solutions. If required, it can be obtained via a software or license key upgrade.

What is Request Rate Shaping?

Individual users may dominate the use of a service, to the detriment of other users of the service. A back-end application infrastructure may have limited scalability, being easily overwhelmed when too many requests are given to it. You may wish to restrict the rate at which certain activities can occur, such as sending an email, or logging in to a service, as part of a wider security policy.

Request Rate Shaping allows you to specify limits on a wide range of events, with very fine grained control over how events are identified. You can impose per-second and per-minute rates on these events.

For example:

You can rate-shape individual web spiders, to stop them overwhelming your web site. Each web spider, from each remote IP address, can be given maximum request rates.

You can throttle individual SMTP connections, or groups of connections from the same client, so that each connection is limited to a maximum number of sent emails per minute.

You may also rate-shape new SMTP connections, so that a remote client can only establish new connections at a particular rate.

You can apply a global rate shape to the number of connections per second that are forwarded to an application.

You can identify individual user's attempts to log in to a service, and then impede any dictionary-based login attacks by restricting each user to a limited number of attempts per minute.

Request Rate Limits are imposed using the TrafficScript `rate.use()` function:

- A virtual server accepts incoming traffic;
- A request rule is run by the virtual server;
- The request rule applies the rate shaping, using the TrafficScript `rate.use()` function.

Note: Request Rate Limits are most commonly used in TrafficScript *request* rules, to shape the rate at which requests are processed. They may be used in response rules if desired, and they can be used to restrict other events, but this is not a common requirement.

Configuring a Request Rate Shaping Class (Rate Class)

To use Request Rate Shaping, you need to add at least one Request Rate Shaping Class, or Rate class. Each Rate class defines a per-second and per-minute rate limit for events.

Rate classes are stored in the Catalog, and are invoked using the TrafficScript `rate.use()` function.

Adding a Rate Class to the Catalog

To add a new Rate class, click **Catalogs**, and then click **Rate** in the menu bar. In the box labeled **Create new Rate class**, enter a name for your new class, and then click **Create Class**.

There are two settings you can configure:

max_rate_per_minute	The maximum number of requests that can take place per minute. '0' means 'no limit'.
max_rate_per_second	The maximum number of requests that can take place per second. '0' means no limit.

You can configure both per-second and per-minute limits if you wish. Both limits are applied (note that if the per-minute limit is more than 60-times the per-second limit, it has no effect).

Once you have configured the Rate class, click **Update** to commit your changes.

Using a Rate Class

Rate classes function as queues. When the TrafficScript `rate.use()` function is called, the connection is suspended and added to the queue that the rate class manages. Connections are then released from the queue according to the per-second and per-minute limits.

Suppose a rate class named 'shape requests' is created, and it is configured with a limit of 10 events (i.e. requests) per second.

The following TrafficScript request rule will only allow 10 requests per second to be processed, no matter how rapidly new requests arrive at the system:

```
rate.use( "shape requests" );
```

As requests arrive, the request rule is executed. Provided that requests arrive at less than 10 per second, the `rate.use()` function just returns immediately and the request is processed without delay.

If more than 10 requests per second arrive, some will be queued by the `rate.use()` function. From a TrafficScript perspective, this function will block. When the request is dequeued, the `rate.use()` function then returns.

The Rate queue

There is no limit to the size of the backlog of queued connections. For example, if 1000 requests arrived in quick succession, 990 of them would be immediately queued. Each second, 10 requests would be released from the front of the queue.

While they are queued, connections may time out or be closed by the remote client. If this happens, they are immediately discarded.

You can use the `rate.getBacklog()` function to discover how many requests are currently queued. If the backlog is too large, you may decide to return an error page to the user rather than risk their connection timing out:

```
if( rate.getBacklog( "shape requests" ) > 100 ) {
   http.redirect( "http://some.other.site/" );
} else {
   rate.use( "shape requests" );
}
```

This rule ensures that no more than 100 requests are ever queued by the rate class.

Avoiding queuing

Alternatively, the TrafficScript function `rate.use.noqueue()` can be used to rate-shape requests in the same manner as `rate.use()`, with the key difference that requests are never queued:

- `rate.use.noQueue()` immediately returns the value 1 if the rate limit has not been exceeded. The connection rate is incremented. This is identical to `rate.use()`.
- `rate.use.noQueue()` returns the value 0 if the rate limit has been exceeded (whereas `rate.use()` would queue the request). You can then decide, using TrafficScript, what you would like to do with the request, such as returning a 503 Too Busy error.

Selective Rate Shaping

You may wish to just rate-limit particular types of request. For example, if you are using a mixture of Java extension pages and other content (such as images, static content etc.) on your web site, you can rate-shape just the Java extension pages:

```
$url = http.getPath();
if( string.endsWith( $url, ".jsp" ) ) {
   rate.use( "shape requests" );
}
```

More fine-grained Rate Shaping

The examples above all apply a global limit. All requests are put into the same rate queue.

In many circumstances, you may need to apply more fine-grained rate-shape limits. For example, imagine a login page; we wish to limit how frequently each individual user can attempt to log in to just 2 attempts per minute.

With a global limit, the effect would be to restrict accesses to the login page to 2 per minute in total. We wish to impose independent per-user limits instead, so that each user is restricted to 2 logins per minute.

It would be possible to create a unique rate shaping class for each user, and select the appropriate class using TrafficScript, but there is a much better solution.

The `rate.use()` function can take an optional 'key' which identifies a new instance of the rate class. This key can be used to create multiple, independent rate classes that share the same limits, but enforce them independently for each individual key:

```
$url = http.getPath();
if( string.endsWith( $url, "login.cgi" ) ) {
   $user = http.getFormParam( "username" );
   rate.use( "login limit", $user );
}
```

The rate shaping limits are applied independently to each different value of $user. As each new user accesses the system, they are limited to 2 requests per minute, independently of all other users who share the "login limit" rate shaping class.

Rate-shaping web spiders

Occasionally, a poorly-written web spider can overwhelm a web site with requests, using excessive bandwidth and reducing the level of service that legitimate site users receive.

The following TrafficScript rule uses a rate class named 'spiders' to limit individual spiders to 1 request per second:

We identify spiders as users who do not prefix their User-Agent header with either 'Mozilla/' or 'Opera/'

We apply limits to each individual spider instance, identified by the user agent and source IP address:

```
$ua = http.getHeader( "User-Agent" );

if( ! string.startsWith( $ua, "Mozilla/" ) &&
    ! string.startsWith( $ua, "Opera/" ) ) {

        $ip = connection.getRemoteAddress();
        rate.use( "spiders", $ua.$ip );

}
```

A more sophisticated rule might only restrict users to 1 request per second if they had previously exceeded a rate of, perhaps 10 per second over a period of 5 seconds, in order to dynamically detect spiders. It could work as follows:

Maintain a list of known offenders (user agent and IP address) using the `data.set()` TrafficScript function. This list would initially be empty.

When a request is received, check the list of known offenders, and use the 1-per-second rate if the request is to be throttled.

Otherwise, use the 10-per-second rate, but before you apply the rate limit, check the backlog. If it is too large (greater than 50 perhaps), then the user has sent too many requests; insert them into the list of offenders.

Graphing Request Rate Shaping

You can use the Activity Monitor to chart the activities of your request rate shaping classes, and you can monitor the values via SNMP.

Note that only global rate limits can be graphed or monitored in this way. You cannot graph or monitor rate limits that are applied to a specific key.

CHAPTER 18 Service Level Monitoring

This chapter explains what Service Level Monitoring is, and how to configure the traffic manager to monitor the response times on your network.

Note: Service Level Monitoring is not available on all Stingray traffic management solutions. If required, it can be obtained via a software or license key upgrade.

What is Service Level Monitoring?

Service response times are a key metric for web services, as lower response times imply a more responsive and usable service. The response time used by Service Level Monitoring is the time in milliseconds taken from the traffic manager receiving a request to then sending the first byte of response data back to the client. It does not include the time taken to return the content to the client on the Internet, and generally is not affected by the speed of the remote network.

Response time includes the time taken for internal processing on the traffic manager (for example, running TrafficScript rules and Java Extensions), the time taken to send the request to the back-end node, and the time taken for the node to generate and return the first data in the response, and the time takes to run any response rules. It is generally independent of the speed or latency of the client connection, so gives an accurate measure of the performance of the internal systems that that application administrator can tune and optimize.

Note: if additional data, such as HTTP POST body data, needs to be read from the client before the node will return a response, the time taken to read this data is included in the response time measurement.

Using Service Level Monitoring, the traffic manager can measure and react to changes in response times for your nodes, by comparing response times to a conformance value. You can obtain a graph of response times and other related information, or issue alerts or log when response times are below the conformance value. The traffic manager also has the ability to dynamically adjust the resources available based on response times, using TrafficScript rules.

Configuring a Service Level Monitoring Class (SLM Class)

To use Service Level Monitoring, you need to add at least one Service Level Monitoring Class, or SLM class. Each SLM class contains the definition of its ideal response time and of thresholds for alerts and logging.

SLM classes are stored in the Catalog, and each can be applied to one or more Virtual Servers.

Adding an SLM Class to the Catalog

To add a new SLM class, click **Catalogs**, and then click **SLM** in the menu bar. In the box labeled **Create new service level monitoring class**, enter a name for your new class, and then click **Create Class**.

There are three settings you can configure:

response_time	Responses that arrive within this time limit are treated as conformant. Responses that take longer than this time are non-conformant. (Time is in milliseconds).
serious_threshold	If the percentage of conformant responses drops below this level, a SERIOUS-level event will be raised, and a SERIOUS-level message written to the event log.
warning_threshold	If the percentage of conformant responses drops below this level, a WARNING-level event will be raised, and a WARNING-level message will be recorded in the event log.

For example, you may set a response time of 1000ms, an alert threshold of 40%, and a logging threshold of 70%. This would mean that:

- If fewer than 70% of requests take less than 1000ms to return from a server, a warning entry will be written in the traffic manager log.
- If fewer than 40% of requests take less than 1000ms, a serious-level entry will be written in the traffic manager event log.

The percentage of conformant requests is calculated from an average of the last 10 seconds' of requests, comparing the number of requests that fell within the

response_time threshold to the total number of requests monitored by that SLM class.

You can define Event Handlers to call actions when warning or serious-level events are raised by an SLM class.

Once you have configured the SLM class, click **Update** to commit your changes.

Applying an SLM Class to a Virtual Server

To apply an SLM class to a Virtual Server, click **Services**, and then click on the **Edit** button next to the appropriate Virtual Server.

Scroll down to **Assigned Classes**, and click **Edit**. Next, click on **Service Level Monitoring** and select the SLM class which you wish to apply.

Click **Update** to apply your changes.

Once your SLM class has been associated with a Virtual Server, it will be used for all requests. You can, however, change the SLM class depending on the nature of the request by using TrafficScript, as follows.

Applying SLM Classes from TrafficScript

TrafficScript includes a number of functions which can be used to read current information from an SLM Class, or change the SLM Class for a connection.

These functions are documented in the TrafficScript Reference guide (available via the online help pages).

Note in particular `connection.setServiceLevelClass()`, which allows you to select a particular SLM Class for the request - overriding the Virtual Server's default - and `slm.conforming()`, which returns the current conformance percentage value.

SLM Class TrafficScript Examples

The following examples illustrate how a default SLM Class can be overridden for certain types of requests, and how to use the performance of one SLM Class to control how you manage traffic.

"FrontPage scripts only" Service Level Monitoring

Microsoft FrontPage™ upload scripts can take a long time to run, particularly when the client is uploading content from a slow connection. This can even lead to a pool being marked as failed, even though there is no malfunction.

To distinguish between response times for normal content, and response times for FrontPage™ binaries, we can use a TrafficScript rule which detects FrontPage™ requests (located in `_vti_bin`).

```
$path = http.getPath();


if( string.StartsWith( $path, "/vti_bin" )
) {

    connection.setServiceLevelClass(
"FrontPage" );

} else {

    connection.setServiceLevelClass(
"Regular" );

}
```

The FrontPage SLM class should allow considerably higher response times than the Regular class used for all other requests managed by the Virtual Server.

Prioritizing resources with Service Level Monitoring

This example tags premium customers with a 'premium' service level monitoring class, and directs them to the 'premium' pool.

Non-premium customers can share the premium pool if the premium SLM class is functioning within its tolerance, but are directed to the 'standard' pool if the premium SLM class is running too slowly.

```
if( $IsPremium ) {

  connection.setServiceLevelClass(
"premium" );

  pool.use( "premium" );
```

```
} else {
  if( slm.conforming( "premium" ) == 100 )
{
    pool.use( "premium" );
  } else {
    pool.use( "standard" );
  }
}
```

Note: `pool.use()` ends the TrafficScript rule immediately and sends the request to the specified Pool.

Graphing SLM Class Conformance Rates

To obtain a graph of data for a specific SLM Class, click on the **Activity** button. Underneath the current activity graph, click on **Change Data**.

The traffic manager now displays a tree of different values which may be plotted. Expand the node marked **Service Level Monitoring Classes**.

Note: You can now select whether to plot data for one particular SLM Class, or for all SLM Classes. You can also plot values for the measurement data used by the SLM Class, for example average response times for the SLM Class. A useful graph might include total connections and total non-conforming connections for all SLM Classes on the same plot, facilitating a visual comparison of the overall performance of a system.

All of the connection data and response time data can be plotted on a per node basis. This can be used to identify nodes which are under performing. A good graph to achieve this would be to plot the mean response times of each of the nodes used by the SLM Class. To select data for a node used by an SLM Class, unfold the section with the name of the SLM Class and check the nodes you are interested in.

CHAPTER 19 Content Caching

This chapter describes the traffic manager's Content Caching capabilities, and explains how it can be configured to reduce the load on the back-end web serving nodes.

Note: Content Caching is not available on all Stingray traffic management solutions. If required, it can be obtained via a software or license key upgrade.

Important: See also the Content Caching section of CHAPTER 20 that discusses the effect on caching when Aptimizer functionality is enabled within your virtual server.

Introduction

Most web servers host a wide range of different types of content, which together make up the web sites and services that are offered. For example, a typical web site will include standard web pages (HTML files), images (GIF, JPEG or PNG files), CSS stylesheets and Javascript source files. Depending on the service offered by the website, the web site is also likely to contain some personalized, custom content that is generated by applications hosted on the web server.

The traffic manager's Content Caching capability allows the traffic manager to identify the content that is not customized for each request and to remember ('cache') the content. The content may be 'static', such as a file on disk on the web server, or it may have been generated by an application running on the web server.

When another client asks for content that the traffic manager has cached in its internal web cache, the traffic manager can return the content directly to the client without having to forward the request to a back-end web server.

This has the effect of reducing the load on the back-end web servers, particularly if the traffic manager has detected that it can cache content generated by complex applications which consume resources on the web server machine.

Configuring Content Caching

Content Caching is performed by an HTTP virtual server. The virtual server stores common responses in a web cache, and replies to common requests by serving the response directly from the web cache.

Applying Content Caching to a Virtual Server

To configure Content Caching for an HTTP Virtual Server, click **Services**, and then click on the **Edit** button next to the appropriate Virtual Server.

Scroll down to **Content Caching**, and click **Edit**. Enable or disable Content Caching for that Virtual Server using the '**webcache!enabled**' option.

Click **Update** to apply your changes.

Configuring lifetimes

The traffic manager will automatically cache the most frequently requested content, so making optimum use of the web cache.

For each virtual server that uses the web cache, you can configure how long resources are cached before they become 'stale' and are discarded:

webcache!time	How long (in seconds) is a resource cached for?
webcache!errorpage_time	How long (in seconds) is an error page cached for?
webcache!refresh_time	How soon before a resource expires should the traffic manager begin refreshing it?

The traffic manager caches regular HTTP responses for the period of time set in **webcache!time**. The resource is loaded into the cache the first time that a remote client requests it; then subsequent requests for the same resource are served directly from the cache until the time expires.

Note: If the resource is changed on the back-end server, it may take up to **webcache!time** seconds before the traffic manager notices and the cached copy is updated. The *Forcing stale content out of the cache* section of CHAPTER 19 describes how content can be forced out of the cache if it has been updated.

The traffic manager also caches error pages (404 Not Found, 410 Gone, 501 Not Implemented, etc), but uses the **webcache!errorpage_time** parameter to determine how long they are cached for before they are requested again from a back-end server. Typically, you would not want to cache error messages for very long; you would want to recheck the content more frequently so that the error message could be purged and replaced with the correct content.

Note: The traffic manager never caches pages returned as a result of a server error, such as '503 Too Busy'.

Expiring resources

When the cached resource expires, it is removed from the cache and the memory can be reused to cache a different resource. If a resource is requested again, it will be retrieved from a back-end server and cached again.

When a resource expires, all requests for that resource will be sent to back-end servers until the complete resource is loaded into the cache again. If there are many concurrent requests for the same resource, or if the resource takes a long time to generate, this can cause a large spike of traffic to the back-end servers.

The **webcache!refresh_time** setting addresses this problem. This value controls how the traffic manager speculatively updates the cache when a resource is due to expire.

For example, suppose the **refresh_time** is set to 10 seconds. In the final 10 seconds before a resource is due to expire, if the traffic manager receives a request for the resource, it will forward the request to the back-end server rather than using the cached response. If the back-end servers provide a valid response, this response is used to update the cache and the expiry counter is reset.

The traffic manager will only send a maximum of one request per second for each resource to the back-end servers during this **refresh_time** period. Other requests for the resource are met using the cached response. In this manner, the cache can be refreshed before it expires, avoiding a large flood of requests to the back-end servers.

Configuring web cache memory usage

It is important to configure the memory usage of the web cache carefully, to ensure that:

- The cache is sufficiently large to contain all variants of the most frequently requested pages;
- The cache is not too large that it causes the host machine to swap part of the running the traffic manager processes to disk.
- All virtual servers share the same global cache memory, and the sizes are configured as part of the **Global Settings** of the traffic manager system.

To configure the web cache memory sizes, click **System**, and then click on the **Global Settings** tab. Open up the **Cache Settings** section of the **Global Settings** page:

webcache!size	Upper limit for the amount of memory (in Mb) that the web cache can consume.

	It can also be specified as a percentage of total system memory.
webcache!max_file_num	Limits the maximum number of entries that can be placed into the cache.
webcache!maxsize	Determines the size of the largest file (in Kb) that will be inserted into the cache.
webcache!verbose	Enable this setting to add a header named 'X-Cache-Info' to HTTP responses to show whether the request/response is cacheable.
webcache!normalize_query	Enables normalization of the query string

Monitoring cache activity

Click **Activity**, and then click on the **Content Cache** tab. This page will display the contents of your web cache, and how many variants of each response are stored. The **Content Cache** page also indicates how much memory is used by your web cache.

If your cache is full, or almost full, either the **Entries** value will be close to the **MaxEntries** value, or the **MemUsed** value will be close to the **MemMaximum** value. In this case, when new entres are inserted into the cache, the least-used entries will be discarded before they expire. You can monitor the value of Oldest to determine the time since the least-used entry in the cache was last used.

You can also click the 'View cache contents' link on the **Virtual Server > Content Cache** page. This will give a summary of cache usage (including the percentage of allocated entries and memory that is currently in use) and a selector so that you can explore the content in the cache and manually expire it if desired.

The Activity Monitor can also be used to graph the cache usage in real-time. Click **Activity** to access the **Activity Monitor** tab. Scroll down and select the 'HTTP Content Cache' report. This cache activity information is also available via SNMP.

Configuring Disk-based Caching

Typically, the traffic manager's cache resides in main memory and the size you select is determined by the quantity of RAM available after accounting for the operating system, application memory, network buffers etc.

As an alternative, the traffic manager may be configured to use a larger cache that is stored on a local disk. This results in an effective two-tier cache because the operating system will store the most active parts of the disk-based cache in memory, using the system's kernel disk buffers.

Configuring the cache

To use a disk-based cache, go to the **System > Global Settings** page and locate the **Cache Settings** section of the page:

1. Configure the size of your cache, in Mb or Gb. This size may be several times larger than the amount of memory in the server.

   If you are running a 32-bit version of the Stingray software, the size of the disk cache will be limited, typically to 2-3Gb. The 64-bit versions do not have this limitation.
2. Enable disk-backed caching with the **webcache!disk** setting.
3. Select a location for the disk cache. This location must be a local directory with sufficient free space, and you are strongly recommended to select a location on fast SSD (Solid State Drive) storage, rather than on a slower local HDD (Hard Disk Drive).

Press **Update**; you will need to restart your Stingray software from the **System > Traffic Managers** page.

The disk cache is not persistent, and is cleared and reset when the Stingray software is restarted. You cannot share disk caches between multiple traffic manager systems.

SSD and HDD disk caches

The disk cache is stored in a single flat file created in the selected location. All reads and writes to the cache can potentially result in a disk access, although the host operating system will cache the most commonly accessed parts of the file and perform writes in the background.

You are very strongly recommended to use a local SSD drive for the cache files because reads, writes and seeks on these drives are very fast. Reads, writes and seeks on a HDD are much slower – a typical mechanical HDD is limited to several hundred disk operations per second, which then limits the number of requests per second the traffic manager can process via the cache.

Caching Policy

Not all web content can be cached. Information in the HTTP request and the HTTP response drives the traffic manager's decisions as to whether or not a request should be served from the web cache, and whether or not a response should be cached.

Requests

Only HTTP GET and HEAD requests are cacheable. All other methods are not cacheable.

The `Cache-Control` header in an HTTP request can force the traffic manager to ignore the web cache and to contact a back-end node instead.

Requests that use HTTP basic-auth are uncacheable.

Responses

The `Cache-Control` header in an HTTP response can indicate that an HTTP response should never be placed in the web cache.

The header can also use the `max-age` value to specify how long the cached object can be cached for. This may cause a response to be cached for less than the configured **webcache!time** parameter.

HTTP responses can use the `Expires` header to control how long to cache the response for. Note that using the `Expires` header is less efficient than using the `max-age` value in the `Cache-Control` response header.

The `Vary` HTTP response header controls how variants of a resource are cached, and which variant is served from the cache in response to a new request.

If a web application wishes to prevent the traffic manager from caching a response, it should add a '`Cache-Control: no-cache`' header to the response.

You can use the global setting **webcache!verbose** if you wish to debug your cache behavior. This setting is found in the **Cache Settings** section of the **System, Global Settings** page. If you enable this setting, the traffic manager will add a header named '`X-Cache-Info`' to the HTTP response to indicate how the cache policy has taken effect.

For further details on HTTP cache policies, please refer to RFC 2616.

Variants

A URL may have multiple variants if:

The response depends on additional request parameters, such as the Accept-Encoding or Accept-Language headers. The origin web server will indicate that this is the case by including a 'Vary' header in the response:

```
Vary: Accept-Language, Accept-Encoding
```

The TrafficScript '`http.cache.setkey()`' function has been used to indicate the response content depends on other computed parameters.

Byte Ranges

Byte-ranges are used by some client software to request a portion of a document. For example, a PDF reader may download the first part of a PDF file (which contains an index of pages), and then issue a byte-range request to retrieve the pages the user wishes to read.

The traffic manager treats byte-ranges as a variant of a URL (see the *Variants* section of CHAPTER 19). Byte range responses are cached, and if another request for the same byte-range is received, the response will be served directly from the cache.

If the traffic manager caches the entire response for a URL, the traffic manager will then serve all byte-range requests from that response.

ETags

ETags (entity tags) are used by web servers to uniquely identify each version of a response. The traffic manager caches and honors ETags and will respond with a 304 Not Modified response if the current cached version of a resource has the same ETag as the version cached by the remote client.

Controlling Content Caching using TrafficScript

HTTP Request processing

When the traffic manager receives a new HTTP request, it must determine whether it can serve a response directly from the web cache, or whether it should forward the request on to a back-end server. The traffic manager executes all TrafficScript request rules on the request before it inspects the web cache. Consequently, a TrafficScript request rule can modify a request and influence the traffic manager's decision.

For example, a TrafficScript request rule could add a '`Cache-Control: no-cache`' header to a request to prevent the traffic manager from serving the response from the local web cache.

HTTP Response processing

Similarly, when the traffic manager receives an HTTP response from a back-end server node, it must decide whether or not to insert the response into the web cache for future use. The traffic manager executes all TrafficScript response rules before making this decision, so a TrafficScript response rule can modify a response and influence this decision.

A TrafficScript response rule could add a '`Cache-Control: no-cache`' header to the response to prevent it from being cached, or a '`Cache-Control: max-age=60`' header to indicate that the response should be cached for 60 seconds.

Note that if a response is served from the web cache, the TrafficScript response rules are not executed.

TrafficScript Cache Control functions

Several cache control functions are available to facilitate the control of the traffic manager's caching behavior. In most cases, these functions eliminate the need to manipulate headers in the HTTP requests and responses.

`http.cache.disable()`

Invoking `http.cache.disable()` in a response rule prevents the traffic manager from caching the response.

`http.cache.enable()`

Invoking `http.cache.enable()` in a response rule reverts the effect of a previous call to `http.cache.disable()`. It causes the traffic manager's default caching logic to take effect.

Note that it is not possible to force the traffic manager to cache a response that it deems to be uncacheable.

`http.cache.setkey()`

The `http.cache.setkey()` function is used to differentiate between different versions of the same request, in much the same way that the `Vary` response header functions. It is used in request rules, but may also be used in response rules.

It is more flexible than the RFC2616 Vary support, because it lets you partition requests on any calculated value – for example, different content based on whether the source address is internal or external, or whether the client's `User-Agent` header indicates an IE or Gecko-based browser.

`http.cache.exists()`

The `http.cache.exists()` function can be used in a request rule to check if a matching response exists at that instant in the cache. If the rule were to terminate

without modifying any of the request parameters, the response would be served from the cache (unless it had expired in the intervening time).

`http.cache.respondIfCached()`

This function can be called in a request rule. If a matching response exists in the web cache, the rule terminates immediately and the response is sent back to the remote client. No further request (or response) rules are run.

If a matching response does not exist, the function returns and rule processing continues.

This can be used to support logic such as rate shaping, whereby you wish to rate-shape requests to back-end nodes, but do not need to apply a rate limit if you can serve the response directly from the traffic manager cache:

```
http.cache.respondIfCached();
rate.use( "my rate limit" );
```

Please refer to the **TrafficScript Manual** for more information and code samples.

Forcing stale content out of the cache

If you update part of the content for your load-balanced web site, the traffic manager will only pick up the new content when the existing cached versions expire.

You can manually force the traffic manager to expire content from the cache, so that it will retrieve the updated versions next time a client requests the content, or you can do so programmatically.

Manual removal of cached content

- Go to the **Activity > Content Cache** page in the Stingray Administration Interface; this page will display a selection of the files that are currently cached by your traffic manager cluster.
- If necessary, use the '**Display Cache Items**' selector to locate the items you want to expire, specifying host header matches and path matches to narrow the selection down.
- Select the entries you wish to expire and press the '**Clear Selected**' button.

This will immediately invalidate the cached copies of these resources, and the traffic manager will retrieve new versions the next time a client requests them.

Programmatic removal of cached content

The Stingray Control API includes SOAP methods to inspect and invalidate entries in the cache; these methods are located in the `System.Cache` interface.

A content provisioning system that uploads new content to the web servers could potentially also invoke a script that used the Stingray Control API to also invalidate any updated items from the content cache.

Please refer to the **Control API** manual for more information on its use.

CHAPTER 20 Aptimizing your web content

Introduction

Faster loading web pages provide users with a more satisfying experience because they reduce waiting time for a web page to render and be usable. Studies show that with an average web page load time of six seconds, only about one second is spent inside the application and web server, with the user waiting another four to five seconds for the client browser to render the web page. Reducing web page load time is vital to improving the overall user experience.

Stingray Aptimizer automatically optimizes web page markup and elements so they load faster for end users. The technology is present within your Stingray infrastructure and no changes are required to your back-end applications.

Note: The Stingray Aptimizer is an optional capability of the traffic manager and is enabled through a specially configured license key. When not licensed, Aptimizer runs in an *evaluation mode*. This mode provides an opportunity to experience the optimization process, in order to evaluate the benefits to your organization without a financial outlay. Contact your support provider for more details of how to purchase a full license.

Configuring Aptimizer for your services

Aptimizer can be enabled as a property of a virtual server, through the **Aptimizer settings** link on the **Virtual Servers > Edit** page of the Admin UI.

It is possible to fully enable or disable Aptimizer functionality for a virtual server using the **aptimize!enabled** configuration key in the Basic Settings section.

A virtual server manages traffic for a specified port and protocol. Requests that it handles may be for a particular application, based upon the URL requested, e.g. http://www.mywebsite.com/supportsystem. This hostname and path combination can be used to create a connection between the application being requested (through this virtual server) and the optimization profile that should be used when it is identified.

In the Aptimizer Profiles section, you can define the mappings between various acceleration *Profiles* you wish to use and the *Scope* within which they should apply. Both of these concepts are **Catalog** objects and must be created before being used on this page. Each of these concepts is explained below.

You can define and map several Scopes to a single Profile, depending on your requirements, though a scope can only be used once within a single virtual server (a *many-to-1* relationship).

The Aptimizer Wizard

As with other Wizards (from the drop-down menu on the navigation bar), Stingray provides an automated process to enable you to quickly set up web content optimization on your web services.

You must have previously created at least one virtual server in order to do this; the wizard cannot run without one. It is recommended that you first follow the **Manage a new service** wizard in order to create a suitable virtual server upon which to run the Aptimizer wizard.

From the *Wizards* drop-down menu, select **Aptimize a web application** to run the wizard.

This process has three main steps, in which you will select:

1) The virtual server upon which you are applying optimization;

2) The Application scope;

3) The Aptimizer profile to be applied for that scope.

Application Scopes

An *Application Scope* object is used to define a collection of URLs that match a specific web site or application hosted by a virtual server. You can find these on the **Catalogs > Aptimizer > Application Scopes catalog** page.

It may be the case that multiple applications are hosted through a particular virtual server, differentiated by the URL in the request. In this case you would set up a Scope entry for each combination of hostname and root path that identify an application, and apply appropriate mappings to optimization profiles on the Aptimizer Settings page.

When a request is received through a virtual server, a match is sought to the most-specific URL within the Scope mappings defined in the Aptimizer Settings. If Stingray finds a match, it applies the appropriate optimization Profile defined in the mapping.

Note: You should ensure that multiple scopes do not provide conflicting URL matches.

The URL can be wild-carded to provide greater flexibility, e.g. *.mywebsite.com, and several hostnames can be included in one Scope. These hosts are checked in the order they appear in the `hostnames` config key list.

The `root` config key can be utilized to further identify the root path to the application in scope.

A built-in scope is automatically provided, entitled "**Any hostname or path**", that can be used where no specific hostname and/or path identification is required within the web application. The hostname field is blank, and the root field is set to '/' accordingly.

Aptimizer Profiles

The aptimizer Profiles catalog contains a list of profiles that can be applied to websites and web applications to enable automatic content optimization. You can find these on the **Catalogs > Aptimizer > Aptimizer Profiles catalog** page.

Built-in and custom profiles

Stingray provides a pair of **built-in** profiles that provide pre-defined optimization settings for your web applications. These profiles are not modifiable or deletable, and are designed to offer a choice of optimization settings applicable to most situations. Choose from:

Basic	This profile is designed to enable a useful number of basic optimization techniques that work well on a wide range of web sites. These techniques include shrinking CSS, in-lining CSS background images, *minifying* JavaScript, shrinking images, URL versioning and advanced caching.
Advanced	This profile enables more advanced optimization techniques in addition to those offered by the Basic profile, including combination of CSS StyleSheets and combination of HTML images into image sprites. These techniques will dramatically reduce page load time, but may require customized rules for compatibility with some sites. In this case, you would clone the profile in order to create a custom Aptimizer profile (discussed below).

If these profiles cannot offer you the desired level of optimization across all elements of your website, you can opt to create a new **custom** Aptimizer Profile that fulfils your specific requirements.

New custom profiles can be created in the *Create a new Aptimizer Profile* section using one of the built-in profiles as a template. Select the desired template profile from the drop-down list and click on the **Create Aptimizer Profile** button. Built-in profiles can also be duplicated to create new custom profiles using the **Save As** option on the *Edit profile* page.

Custom profiles provide a range of additional acceleration options, as discussed in the Understanding custom acceleration profiles section below.

Optimization modes and page information

Aptimizer can run in one of three modes, defined using the `aptimizer_mode` configuration key:

- **Off**
  Acceleration is disabled, but requests for Aptimizer resources are served.

- **Stealth**
  Acceleration is disabled, except where enabled using the query string command:
  `?aptimizer=on`

- **On**
  Acceleration is enabled for all requests.

You can elect to show request information for each web page that is passed through an Aptimizer Profile using the `aptimizer_show_info_bar` configuration setting on the Edit Profile page. If this is set to **Yes** and acceleration is currently active, all pages in this session will show a status bar which includes controls to provide information specific for that request.

Measuring Aptimizer changes

This section explains how to test Aptimizer and measure web page performance using a variety of tools. These measurements can be used to determine how effectively Aptimizer is accelerating your web service.

Checking that Aptimizer is active

To easily check that your web content is being accelerated, use your browser's "View HTML Source" option after loading a web page. Search for this comment, where `x.x` is the Stingray version used:

```
<!-- Optimized for speed - Stingray Aptimizer x.x - riverbed.com/aptimizer -->
```

If you don't see this comment, please refer to the Configuring Aptimizer for your services section above.

Using Stealth Mode to test Aptimizer

If you do not have a development or test environment available you can still test Stingray Aptimizer by running the accelerator in Stealth Mode.

In Stealth Mode, Aptimizer will not apply any optimization to web pages, unless commands are passed via the query string. End users will not see any change in the way web pages behave or appear.

When running Aptimizer in Stealth Mode all incoming requests for pages and objects will be served in their original form. If the request contains the parameter `aptimizer=on` in its query string, Aptimizer will send a cookie that represents an explicit instruction to accelerate all requests for this browser session.

When Aptimizer sees the enabling cookie on the incoming requests it will intercept and accelerate them according to the rules and settings defined in the Aptimizer profile applicable to that scope.

These are examples of valid requests:

```
www.example.com?aptimizer=on
www.example.com/destinations?aptimizer=on
```

You configure Aptimizer to run in Stealth Mode by changing the **aptimizer_mode** configuration key in the relevant profile.

Measuring web page speed

To find exactly what has changed since enabling Aptimizer you will need before and after web page load time measurements.

An initial measurement should be used as a baseline to which further measurements will be compared. Ideally the baseline should be documented while the web site is not yet running with Aptimizer enabled, or with Aptimizer running in Stealth Mode.

At each stage a new set of measurements can be taken and used for comparison with the baseline. The stages could be any milestone in the Aptimizer deployment, for instance:

- Initial deployment with default settings
- Changes to rule options
- Changes to CSS settings
- Changes to caching behavior

- Changes to default image sets

Tools

There are many tools available to verify web page load times and provide feedback on web page structure. Some can be used in a very granular fashion, allowing testing on individual web pages. Others are used for collecting metrics that can be correlated to the Aptimizer-enabled implementation.

Page speed and structure

Many tools can test and provide information about your web pages. Some are available as online services while some are available as an add-in for web browsers such as Firefox or Chrome. Some typical examples follow, however your exact requirements may vary:

- WEBPAGETEST (www.webpagetest.org) is a free web site service that emulates browser behavior and retrieves your web page for analysis, presenting results in a cascade-style chart showing how a browser progresses through the page when rendering it.

  A simple process for ensuring that the site is as fast as possible is to work your way down the stepped list of objects on the chart, determining the most appropriate acceleration to apply for each object. For example, the start of the chart might show two redirects and then the page loading. The ideal action for this would be to rework the site such that the redirect is not used. Or you may be looking further down the chart at a script that loads on the page.

- YSlow (http://developer.yahoo.com/yslow/help) is a browser add-in developed by the Yahoo! Developer Network initiative. The software uses built-in rules to grade a web page across a range of performance indicators, from A (best) through F (worst) and provide suggestions to improve the web page score. It runs on Firefox and Chrome browsers.

- Google Page Speed (http://code.google.com/speed/page-speed) is another browser add-in also available for Firefox and Chrome browsers. The software grades web pages on a Page Speed Score from 0 (worst) to 100 (best). Google Page Speed also provides suggestions on how to improve the web page score.

- Riverbed makes the Aptimize Site Analyzer tool available to customers and partners. The tool runs on a Windows-based PC to load and analyze a web page, providing a cascade-style chart and timings all in a single tool. It also generates a PDF file with summary findings.

Other metrics

Some metrics require a longer data collection period to be useful. This means collection must start even before Aptimizer is enabled.

- Google Analytics (www.google.com/analytics) is a free tool that can be used to collect metrics such as page load time, number of page views, page views per visits and bounce rate. Google Analytics requires a small script to be added to all pages in your web site. Results are visible a couple of days after installing the script, but you need to collect data for some time to allow for effective comparisons.

- Google Webmasters (www.google.com/webmasters) is another free tool that can be used to manage how Google sees your web site. As a bonus it provides a chart showing the average global web page load time for your entire web site, over a rolling window of six months. Google Webmasters requires site ownership authentication, which can be accomplished by a small DNS change, a configuration file added to your web server, or a meta tag added to your default web page.

Understanding custom acceleration profiles

Typically, the built-in Aptimizer acceleration profiles provide good general-purpose optimization for your web applications. However, there might be times when you need to define *custom* acceleration profiles in order to provide a more fine-grained control over optimization of specific elements of your web site or application. You can find an overview of profiles in the Aptimizer Profiles section above.

The Edit page for a custom acceleration profile includes an additional control referred to as the **Acceleration Settings** dialog.

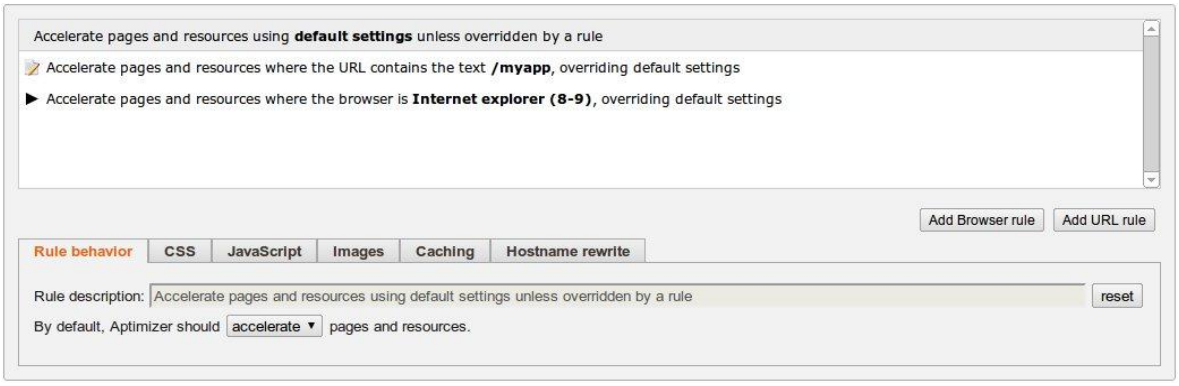


Fig. 34. The custom profile accelerations setting dialog

This contains controls and rules to handle optimization of different content types within your web application. It is split into two main parts:

- The top part shows the hierarchical list of rules that are applied to your web content, usually based on URL location or browser type;

- The set of tabs below this represent the properties that apply to the current rule. As you highlight a particular rule, its properties are displayed in the tab set below it.

A **default settings** rule is provided for each Acceleration Profile that cannot be removed. This allows you to define the default behavior of the profile and configure the fall-back values for each of the property tabs. New rules can be added beneath this to further define optimization behaviour and to override the default properties based on either a particular browser type (click on **Add Browser rule**) or a URL (click on **Add URL rule**). These newly added rules can be ordered according to priority or desired effect by clicking on the up or down arrows to the right of each rule bar. They can be removed altogether by clicking the delete button to the right.

Note: Each Browser rule provides the ability to add sub-rules based upon URL, in order to provide a combination of both rule types. Click on the **Add URL rule** button within the currently highlighted Browser rule, to the right of the rule description, to append a new URL rule below it. The hierarchical dependency of this new rule is shown by it being indented in the list below the parent Browser rule.

In the rule properties section of the dialog, each displayed tab contains settings relevant to content that constitutes a typical web application, such as CSS, Javascript and images. The first tab, entitled '*Rule behavior*', contains the general description of this particular rule and governs it's overall behavior within the scope of whether to accelerate or exclude the content defined by the rule.

Overriding default behavior

Each new browser or URL rule added to the list has its own set of properties that can override the default behavior when a match is made. Each property tab contains an **Override** checkbox that enables or disables this mechanism for that particular property tab. Should you wish to provide an alternate configuration setting for a particular rule, click the Override checkbox to enable the settings for that tab. By default, all settings are disabled/greyed-out until the override checkbox is selected.

Rule chaining

The default behavior in any given situation is for each request to pass through the list of URL rules and stop when a match is found. No further URL rules are tested, and the request is then passed through the list of browser rules to identify a match in this category. Should a matched browser rule have one or more URL sub-rules attached to it, a request will be subsequently tested against these and stop if a further match is identified. This *stop-when-matched* behavior can be overridden by selecting the **Allow settings for this rule to chain to other rules** checkbox within a URL rule.

If enabled, a matching request will not stop at that URL rule. Instead it will continue on to find a second rule match in that logical block (the list of URL rules at the same hierarchical level). Note, however, that this setting only remains in effect until a further match is found. At that point, if the second matching rule does not also have chaining enabled, the request will not attempt to match any further rules. If you wished to have all URL rules tested throughout your profile, regardless of how many matches occur, you would need to enable chaining on each of them.

Acceleration settings

Each group of related optimization settings are shown on a separate tab. These are listed below.

Rule behaviour tab

Rule Description	A textual description of the current rule, automatically assembled from the basic rule behavior settings contained in this tab. For example, changes to the URL test string or Rule mode will be automatically reflected in the descriptive text in this field. It is possible to override this text with your own description, however no further automatic updates will be made. To reset the description back to the default auto-generated string, click the **reset** button.
Rule mode	Aptimizer can be set to **accelerate** or **exclude** all pages and resources identified by this rule. Accelerate mode is the default, and performs the desired optimization defined within the rule properties on a matching request. The alternative is to set the rule to exclude optimization for pages or resources that match the URL or browser type. Note that browser rules can be overridden further by adding a URL sub-rule within it that is set to Accelerate. URL rules have a third mode, **strip**, that strips out resource references where a match is encountered.
URL test	(URL rules only) These rules only apply to requests that match the specified text. The text is compared with the full URL including the hostname if the **Match external**

	resources box is checked, otherwise it is compared only with the URL path. You can specify a substring that must be contained in the URL, or a wildcard or regular expression that must match the entire URL.
Applies to externally hosted resources	(URL rules only) This setting allows you match against resources held external to the web application. If enabled, the rule will only match requests for hosts that are not included in the Application Scope, and the string contained in the **URL test** field will be matched against the hostname and path contained in the request. For example, you may wish to exclude media files on your web page that are included from an external media hosting website. If this is the case, you can select this option and specify a wild-carded string in the **URL test** field that includes the name of the hosting site in addition to the URL path of the media files you are interested in.
Rule chaining	(URL rules only) Allow settings for this rule to chain to other rules. Please refer to the Rule chaining section above for more details.
Browser list	(Browser rules only) This operation allows you to build up a list of browser types and specific versions in order to narrow the match parameters for the rule. Specify the browser type and, optionally, the minimum and maximum applicable version numbers. Click **Add** to add it to the list.

CSS tab

CSS and Stylesheet optimization	**None** – Style sheets will remain in their original form wherever they appear. **Optimize CSS Content** – Style sheet blocks in the page and in external style sheets will have all redundant

	whitespace and comments removed. **…and combine consecutive stylesheets** – Style sheets that appear as consecutive references (i.e. without inline style blocks or excluded style sheet links between) will be combined and each combined reference inserted to the location where the first style sheet in a set appears in the document. **…and combine all stylesheets** – All style sheets will be combined, no matter where they appear in the page. The new combined reference will be inserted at the top of the head section of the document.
CSS background image optimization	The maximum allowable size (in bytes) of base64 encoded data to be inlined using data:uri notation or a MHTML reference. Images that exceed this limit will remain as URLs.

JavaScript tab

JavaScript optimization	**Enabled** – Script files will have all redundant whitespace and comments removed. **Disabled** – Script files will remain in their original form.

Images tab

Combine images	**Enabled** – All images that appear as <img> tags in the HTML will be combined into an image sprite according to the image format. **Disabled** – Images that appear as <img> tags in the HTML will not be combined into image sprites.
Shrink images	Images will be shrunk where possible, removing unnecessary information such as metadata, redundant

	color information and in the case of JPEG images, reducing the quality to reduce the size of the files.
Resample JPEG quality	The percentage quality level to resample JPEG images to (0-100). If retaining the quality of your JPEG images is paramount, then set this value to 100. The default is 85.
Convert all images	Provides the ability to convert all images from their original format to same new format as selected in the drop-down box, before being added to an image sprite or served. Choose from JPEG, PNG or GIF. If an image appears on a web page in BMP format, it will automatically be converted to JPEG format by Stingray to reduce its size. This can be prevented by excluding the image from processing with a rule.

Caching tab

Version resource URLs	**Enabled** – URLs at this location will be versioned by appending a query string parameter that is a hash of the contents of the file. The cache duration used for versioned URLs is the greater of R***esource Client Cache Duration*** or 90 days. **Disabled** – URLs will be left unaltered and clients will cache the item for the full R***esource Client Cache Duration*** interval before rechecking.
Mark resources as cacheable by intermediary proxies	**Enabled** – Resources that are cacheable at this location will be marked with the *Cache-Control* public header which allows them to be cached by intermediary proxy servers. **Disabled** – Resources that are cacheable will be marked

	with the *Cache-Control* private header to prevent them from being cached by intermediary proxy servers.
Optimize and cache resources	**Enabled** – If a resource such as an image or CSS stylesheet is requested by its original URL, enabling this option will cause Aptimizer to optimize and cache the resource before returning it to the client. This can be useful if, for example, your site contains JavaScript that fetches images or other resources from your server after the page has loaded. **Disabled** – Directly requested resources are returned un-optimized and will not be cached by the traffic manager.
Cache on browser	The duration, in seconds, that resources will be cached on the client's browser before expiring. This is used as a sliding expiry to set the *Expires* header, meaning the client will cache the file for the specified number of seconds from when it receives it.
Background recheck	The time, in seconds, after which a resource that is in the server cache will be marked for a recheck. When a resource that is marked this way is requested, the existing version in cache will be returned immediately, while a background task refreshes the cached version if it has been modified.

Hostname rewrite tab

URL hostname rewriting	Relocates resources from the current domain onto the specified host or URL root by rewriting all matching URLs. This can be specified as a single hostname (e.g. cdn.example.com) or a root URL (e.g. https://cdn.example.com/images).

Understanding optimization techniques

This section discusses various optimization principles employed by Stingray Aptimizer. The first part relates to the concept of *web page speed rules* and how Stingray Aptimizer applies those rules to optimize your web content. We then discuss a number of techniques Aptimizer uses to apply optimization to web applications.

Web page speed rules

These broadly fall into four categories. Each of which can be applied to your web application in order to provide a better response to the client browser.

- Reduce the number of objects being loaded by web pages;
- Reduce the size of everything sent to and from the server;
- Cache as much as possible to speed up repeat views;
- Reduce the time it takes for the server to respond to a request.

The sections that follow describe each in more detail.

Reduce the number of objects being loaded by web pages

For each element on your web page, the browser will issue a HTTP request to the host where the resource is stored. Browsers limit the number of simultaneous connections that can be established to each server; therefore fewer web page elements mean a faster web page load. It is this time optimization step that will accelerate web page loading the most.

Aptimizer automatically reduces the number of objects by combining files when appropriate, creating **combined resource sets** by resource type. Specific information regarding the most common types is shown below:

Image in-lining and combination

Images are important elements in every web page. They can be separated into CSS background and HTML images, with different approaches for each type:

- **CSS background images**

  CSS background images are referenced from the background property in a custom style sheet or directly in a style property within a web page. To reduce the number of requests for these resources, Aptimizer will convert them to base64 data and embed this in the style sheet file or HTML.

  The exact acceleration behavior differs depending upon the browser:

  - For Internet Explorer (version 6 and 7) the base64 data is served in a separate file, with the style sheet background URL referencing this file using MHTML notation;
  - For all other browsers, the base64 data is embedded directly into the CSS using the *data:uri* notation.

- **HTML images**

  HTML images are images referenced from an `<img>` tag in a page. To accelerate these images, Aptimizer can create a combined CSS sprite that contains all images of a particular format.

  Aptimizer then attaches some CSS style information to the `<img>` tag to reference this sprite as a background image, and sets the *source attribute* to reference a transparent GIF.

JavaScript minification

Reducing the amount of superfluous content in JavaScript files run from within the HTML can help to improve overall page load time. Content such as redundant whitespace and comments, whilst useful from a software development perspective, serve no functional purpose in the operation of the page so are removed in order to reduce the file size.

Style sheet combination

Reducing the style sheet request count is an important step in accelerating the time a browser starts rendering a web page. Combining style sheets is generally a safe and effective way of accelerating page load time.

Accelerating third party domain content

By default, Aptimizer will apply web page acceleration rules to elements served from the same web sub-domain from which the page is served. Consider the example of an image URL in the `<img>` tag for the web site `www.example.com`:

URL	Result
/images/sample1.jpg	Aptimizer accelerated
www.example.com/images/sample1.jpg	Aptimizer accelerated
images.example.com/sample1.jpg	not accelerated
images.otherexample.com/sample1.jpg	not accelerated

It is possible to modify this configuration to include other web sub-domains, even third party domains, effectively accelerating partners' content required to display your web pages.

Aptimizer can rewrite these resource references to either:

- combine the object into a CSS sprite set, or
- apply versioning (see the Resource naming and URL versioning section below) to the original URL, which will be served from the domain host.

A technique to increase the number of concurrent resource downloads is "domain sharding", where some of these resources are downloaded from different web domains or sub-domains. This approach provides more benefits than domain sharding, because every distinct web domain name used on a web page requires additional DNS resolution time and TCP connection establishment time. Also, the chance of some content being unavailable is greatly reduced, since this content is now being served from your web server.

Reduce the size of everything sent to and from the server

By default, Aptimizer works to automatically reduce the size of content sent from the server.

- Text based objects are compressed using the deflate compression algorithm (or gzip if the browser does not support deflate);
- JavaScript and style sheet files are minified by removing all redundant whitespace characters and comments;
- Redundant image information (such as metadata) is removed, and JPEG images are, by default, resampled to 85% of their original quality.

Cache as much as possible to speed up repeat views

Browsers can locally cache resources for faster reuse on subsequent web page visits.

Aptimizer automatically applies cache headers to all resources, providing browsers with information needed to manage those resources. By default, Aptimizer applies different cache durations to resources depending on how they are referenced, as shown in the following table:

Condition	Cache for
Object is combined into a resource set	1 year
Object URL was versioned by Aptimizer	90 days
Original object URL was requested	1 hour

Public proxy caching

Optionally, Aptimizer can help leverage any intermediary caching proxy server, such as those commonly used in corporate environments and internet service providers (ISPs). By using a specific attribute, Aptimizer will add "`cache-control: public`" headers to all or any specific resources.

Marking resources as proxy cacheable can provide a significant improvement in both first view and repeat view load times for users who access a web page on a web site other users have already accessed.

Reduce the time it takes for the server to respond to a request

This rule applies to the time it takes for your server to start sending a HTML response to your client's browser. As previously mentioned, this can be up to 20% of the web page load time.

Reducing this time is directly related to your server, application design and maintenance. Possible solutions can involve scaling up (achieved by adding system resources such as processor and memory to existing servers), scaling out (achieved by increasing the number of server nodes to distribute workload over more nodes), code profiling and tuning, database tuning, sharding (distributing the database over multiple servers), web server tuning and more.

This rule applies to your web server and application, thus outside the scope of this chapter.

Resource naming and URL versioning

Aptimizer will create a new identity for each **combined resource set** or **URL auto versioning** created in the process of accelerating your web pages. This approach allows Stingray to inform the browser when a resource has changed rather than telling it to check back with the traffic manager every so often to see if it has changed. We do this by making the URL of the resource dependent on its content.

For example, a resource set could be named:

```
/aptimized-gif-kjmMLxSEdXYqXt3coAasow4F=-ds3432d.gif
```

and an auto-versioned URL could look like:

```
/images/aptimized-jpg-23dgdsds5563gfv=-P9vry3Uq22.jpg
```

The string of numbers and letters in these URLs represent a unique hash value of the settings, the content and any child resources that make up a particular resource. Whenever a setting is modified, any resources for which that setting applies will automatically be recalculated. Thus, any change to the content of a resource will also cause this unique value to change.

Under normal circumstances, where the traffic manager is simply serving pages without auto-versioning enabled, the following typical data flow is in operation:

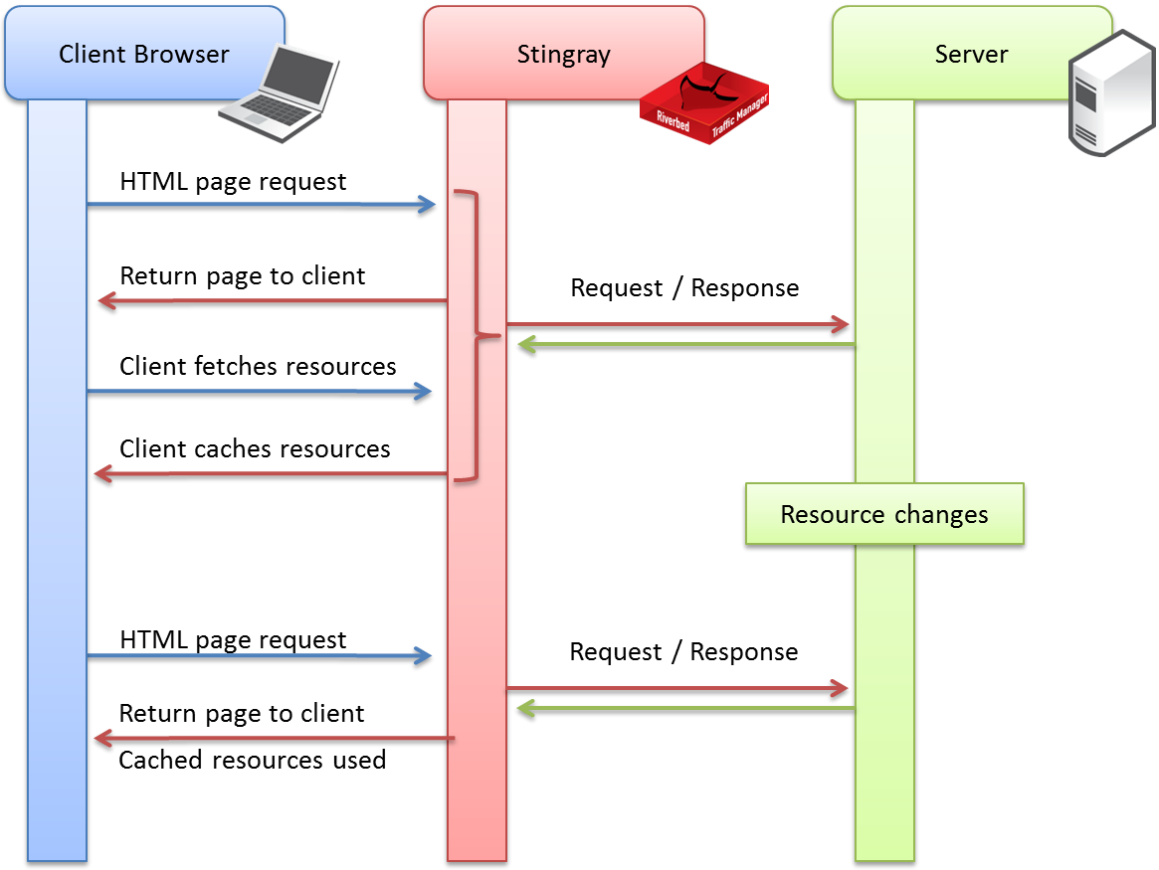


Fig. 35. A typical data flow with auto-versioning disabled

- The client requests an HTML page through a service running on Stingray;
- The client receives the HTML page back, with links to regular resources (for example, style sheets, images, etc.);
- The client fetches these resources;
- Stingray serves them from the back-end server, possibly optimised depending on the Aptimizer settings used;
- The client caches any available resources locally;
- Some resource is changed on the back-end server for whatever reason (for example, bug fixing, a layout change, etc.);
- The client requests another HTML page on the same website;
- The client receives the HTML page from Stingray with a link to the same resources;

- The client is unaware of the change on the server, and refers to its own locally cached (and unexpired) versions of any identified resources and does not re-request them from the web service.

As shown in the last step, one essential flaw with this scenario is that the client will wait a period of time before re-requesting a cached resource, which may or may not have changed. The client could therefore be using an out of date copy of a resource until some event occurs to prompt the re-request.

With auto-versioning enabled, Stingray provides the assurance that the client will always be using the most up-to-date copy of a resource contained in a web page. The following data flow demonstrates this:

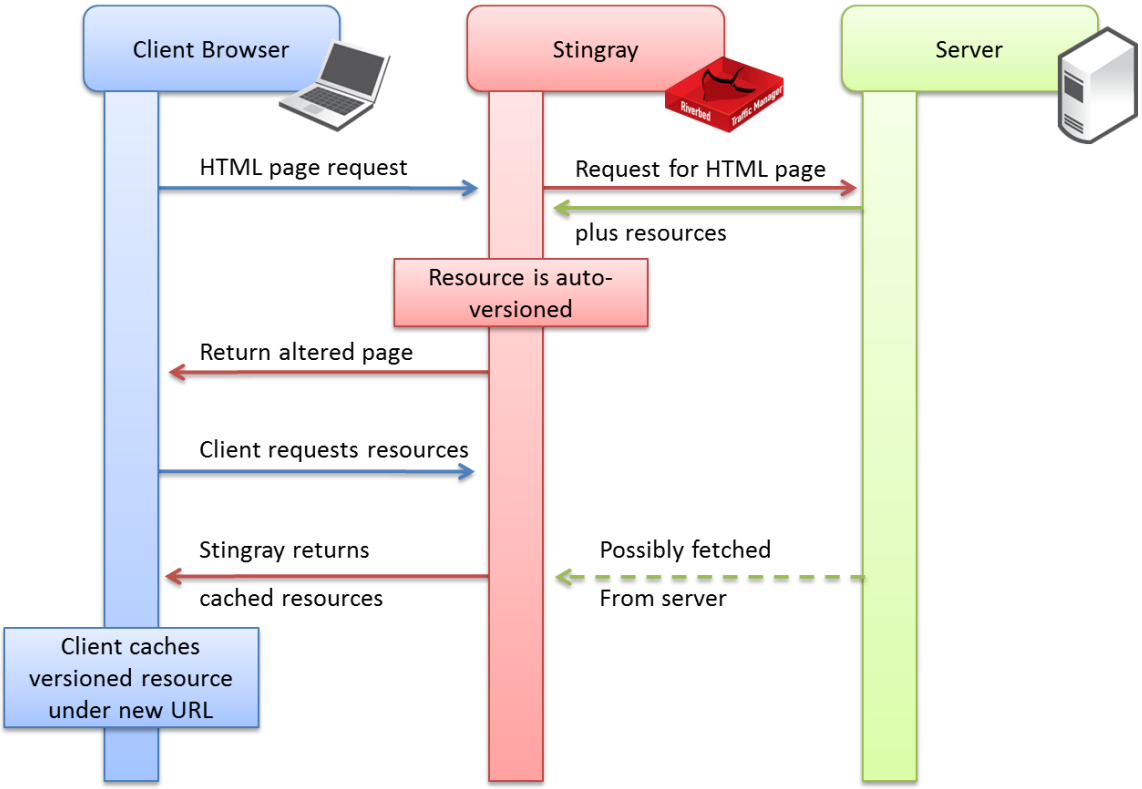


Fig. 36. Typical data flow with auto-versioning enabled

- The client requests an HTML page from a web service provided through Stingray;

- Stingray notices the HTML page references one or more resources;
- Stingray fetches these resources (and possibly optimises them, depending on the Aptimizer settings), examines the content and creates a hash value based upon it;

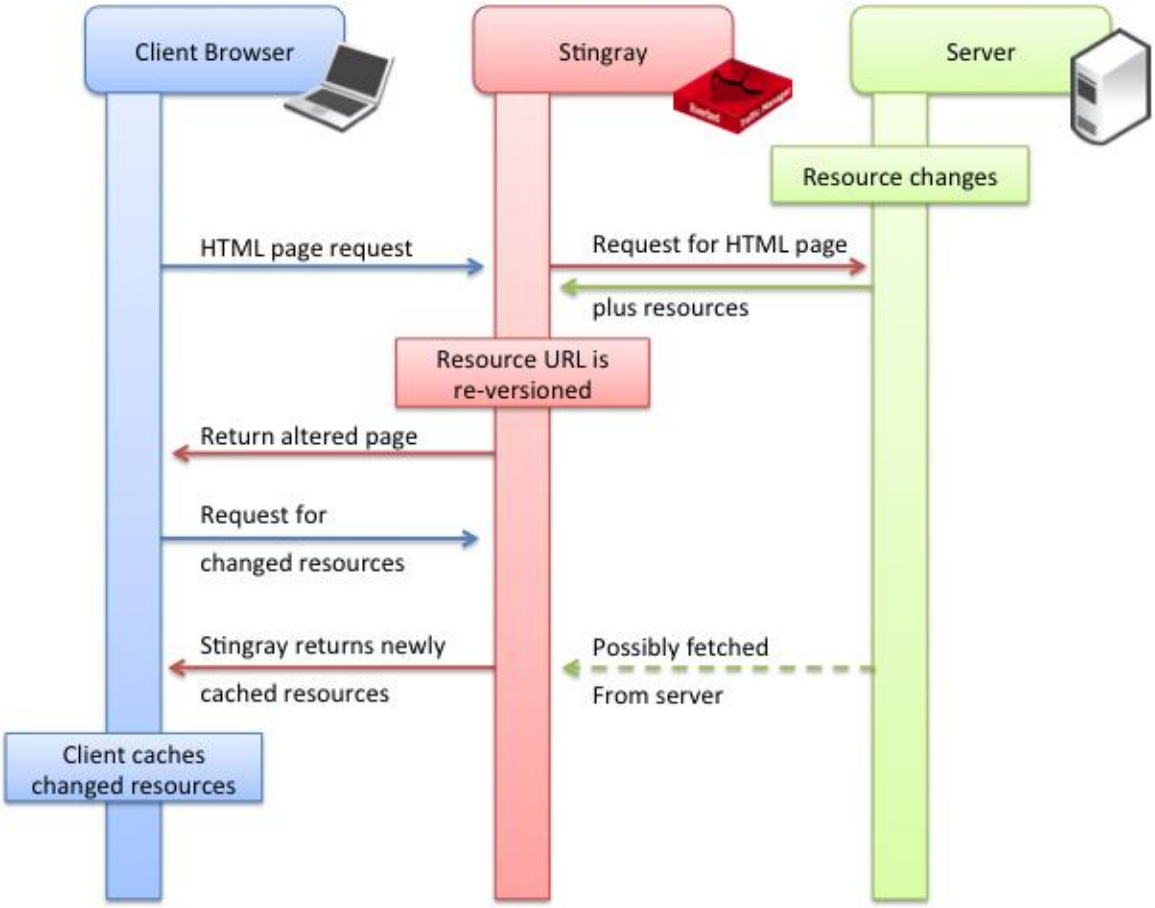
- Stingray creates a new URL for such resources that includes the computed hash value, and replaces all references in the HTML page;

- The altered page is returned to the client;
- The client requests the in-page resources from Stingray, via the new URL;
- Stingray serves the identified content from its cache (or fetches it from the back-end server if it's not in the cache);
- The client locally caches resources using the new URL;

Subsequent page requests are then served from Stingray with the test that the computed hash value of each resource hasn't changed (based on the resource content at the back-end):

- The client requests another HTML page;
- Stingray observes that the contained resources haven't changed on the back-end server and writes in the same computed hash-value URL as last time;
- The client uses it's cached copies of those resources;

Should a resource change on the back-end server, subsequent page requests will be affected by virtue of the computed hash-value being different. The following scenario occurs:



- A resource change forces a re-compute of the content-based URLOne or more resources change on the back-end server;
- The client requests another HTML page;
- Stingray observes that the desired resource has changed on the back-end server and creates a new URL based on the new content - different to the previous one;
- The client doesn't have a cache entry for the new URL, so it requests it from Stingray, which serves it back;
- The client is immediately using the new version of any changed resource.

Using a Content Distribution Network

Content Distribution (or Delivery) Networks (CDNs) host some or all of your resources on servers that are geographically distributed, reducing latency between your web server and the customer. For further information on CDNs, please refer to http://en.wikipedia.org/wiki/Content_distribution_network.

You can leverage a CDN with Stingray Aptimizer using the **URL rewrite to CDN** configuration setting on the Aptimizer Profiles edit page (custom profiles only).

This setting instructs Stingray to rewrite the HTML for requests that match that rule so that resources are then downloaded from the domain provided by your CDN instead of your own web server.

Stingray will automatically use the scheme (HTTP/HTTPS) of the incoming request when accessing a CDN host, but it is possible to manually specify the scheme and virtual path in the **URL rewrite to CDN** setting if required. This allows the scenario of switching all traffic to SSL if the target web site is reverse proxied through another device that terminates the SSL connection.

Troubleshooting Aptimizer

Interaction with other Stingray functionality

In practise, enabling Aptimizer on your virtual server should present no interoperability issues with other Stingray functionality. However, due to the nature of certain built-in protection and performance capabilities, it is important to understand the interactions between Aptimizer and these other features.

Problems with dependent requests

The following diagrams demonstrate the consequences of what are termed *dependent requests*. These are requests that follow a web page request handled by the traffic manager, for dependent resources contained within that returned page.

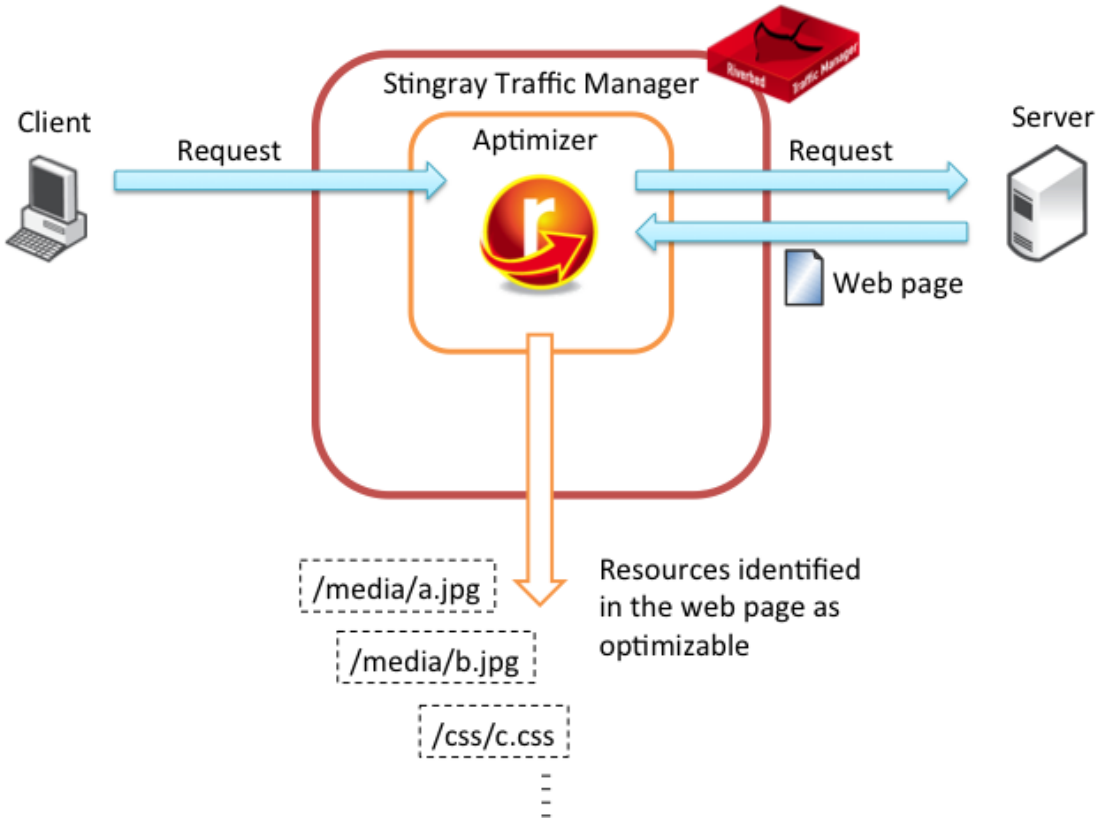


Fig. 37. Identification of optimizable resources

Initially, the client requests a web page for a service handled by the traffic manager. Aptimizer is enabled for this web application and will attempt to identify resources that are potentially optimizable.

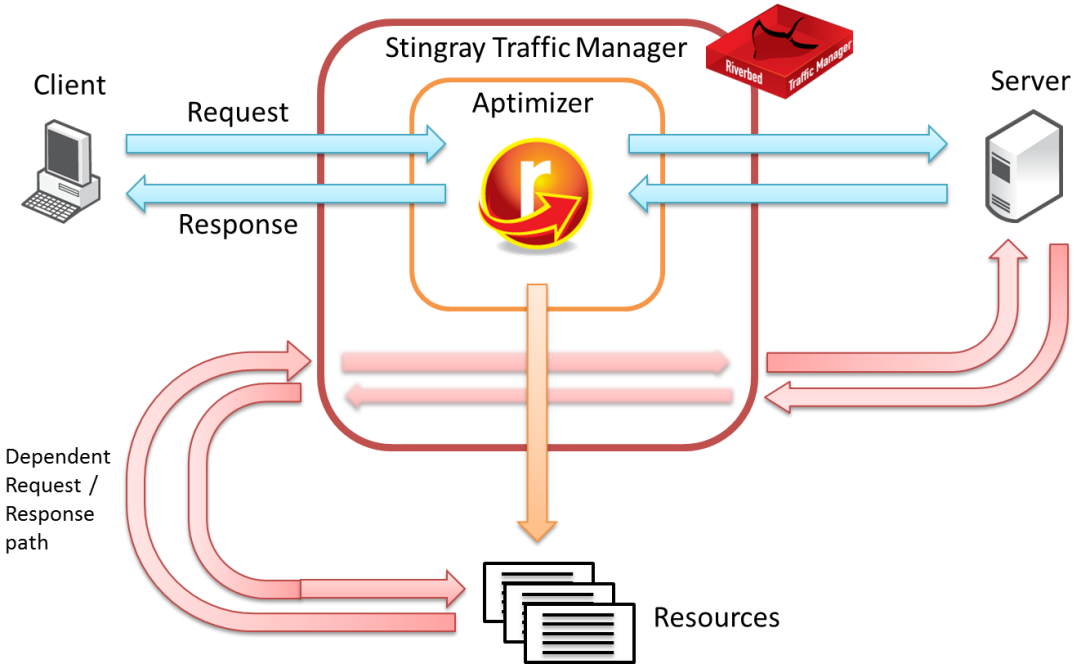


Fig. 38. Dependent request/response model

The traffic manager may then need to make a number of additional *dependent requests* in order to locate and provide the various resources that have been stripped or optimized out in order to improve the page load time of your application.

Dependent requests are processed through the Traffic Manager as typical requests, with the usual processing applied. However these requests do not necessarily contain the same identifying headers, cookies and other elements of authentication as the requests and responses made directly through the Aptimizer, so can cause potentially unexpected results.

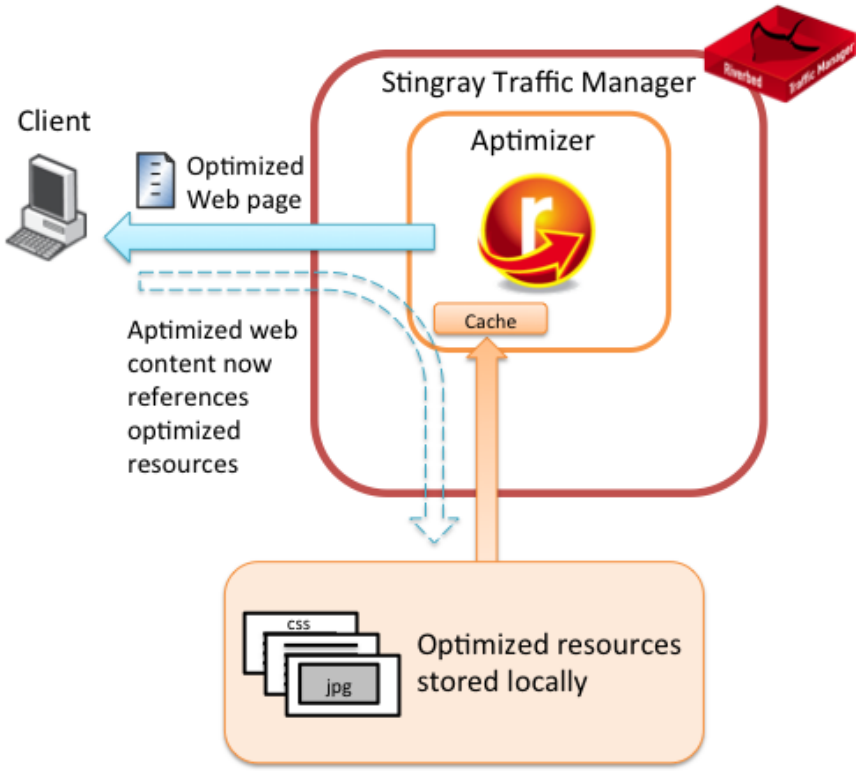


Fig. 39. Optmized resources are cached and served by Aptimizer

At this stage, suitable resources are optimized and cached locally by Aptimizer. Subsequent page requests contain modified URLs to optimized content, which can be served directly from the traffic manager.

The following sections highlight the issues surrounding the fetching of these dependent requests.

Service Protection

The fetching of dependent requests could in some cases trigger Service Protection classes to be activated (since the requests will come in bursts from one of the traffic managers in your cluster). This can be prevented by adding the appropriate traffic manager IP addresses to the list of **Allowed IPs** (also known as *whitelisting*)

maintained within the **Access Restrictions** section of the class configuration. See CHAPTER 15 for further details of Service Protection features and capabilities.

TrafficScript Rules

TrafficScript rules that rely on the client IP address, browser headers, cookies, authentication data, or any other information pertaining to the client, could result in unexpected behaviour due to the lack of such information in dependent requests.

SSL Decryption

Stingray cannot be used to aptimize virtual servers that use SSL decryption and require client certs, since dependent requests cannot pass themselves off as the real client (for lack of the correct private key).

Content Caching

Web content that is passed through the Aptimizer and can be considered to be in scope with Aptimizer will not be cached in the usual way within the regular Stingray web cache. Instead, it will be held and served from a dedicated Aptimizer cache, designed to maintain local copies of optimized web content/resources. This behavior is controlled via the **Caching** tab on the Aptimizer Profile edit page.

Run-time errors

Runtime errors are generally server-related. You will get these errors due to server misconfiguration or third party interference (routers, firewalls, load balancers). In general you won't be able to access the Web site at all, of if you can access the web site you will notice missing resources.

Symptom: An error page is displayed only when Aptimizer is enabled; resources are served with a 400 or 404 http status; the server takes a long time to return web pages even though the application pool is warm.

> **Problem**: A URL rewriter or other HttpModule is attempting to handle the Aptimizer resource set URLs.
>
> **Resolution**: Exclude URLs containing `/aptimized-gif-*.gif` from URL rewriting or processing by another HttpModule.

Symptom: The web service works when accessed locally but not when accessed from external clients.

> **Problem**: A proxy or upstream device is blocking or altering `/aptimized-gif-*.gif` URLs causing them to become invalid.
>
> **Resolution**: Allow requests for `/aptimized-*` to pass through to the host server unmodified.

Symptom: Cannot see the comment "`<!-- Optimized for speed - Stingray Aptimizer x.x – riverbed.com/aptimizer -->`" in the HTML (where `x.x` is the Stingray version being used).

> **Problem**: When you use your browser's "View HTML Source" the comment "`<!-- Optimized for speed - Stingray Aptimizer x.x – riverbed.com/aptimizer -->`" is not present, indicating the web page is not being accelerated.
>
> **Resolution**:
>
> - Make sure Aptimizer is enabled for the specific virtual server.
> - Make sure the URL is not being excluded by a rule set.
> - If Aptimizer is running in Stealth Mode use the `?aptimizer=on` query string in the URL request.
> - If the URL is not being excluded, Aptimizer is fully enabled, make sure `?aptimizer=off` is not used in the URL request.

Image errors

You will have image errors if your web pages do not display images in their specified position or display images with incorrect size. Image errors can also be "invisible", as it is the case when Aptimizer creates new resource sets for every new visit to the page.

You can spot the problem with resource set creation by inspecting the elements on the page. You should see the most common elements being created once and served from the cache multiple times.

Symptom: A new sprite set seems to be created every time I visit the page.

> **Problem:** The list of images in the page is different every time, which causes Aptimizer to create a new sprite set every page visit. This can cause excessive processor utilization, high cache memory usage, and overall reduced performance on the server.
>
> **Resolution**:
>
> - Add a URL rule to include the dynamic images in an uncombined form (disable the setting **Combine HTML images into image sprites** in the **Images** tab).

Symptom: Images on the page are losing their padding or out of alignment on the page.

Problem: Padding is being used to space the image from its neighboring UI elements

Resolution:

- Add a URL rule to include the dynamic images in an uncombined form (eg. Disable the setting **Combine HTML images into image sprites** in the **Images** tab).

Symptom: One of the images appears much larger when Aptimizer is enabled.

Problem: The container that holds the image uses style sheet to resize the image from its native dimensions.

Resolutions:

- Modify the page template to include a width and height attribute on the <img> element. This lets Aptimizer know what size the image should be displayed at, and also helps the browser calculate the page layout more quickly, resulting in a faster rendering time.
- Add a URL rule to include the dynamic images in an uncombined form (eg. Disable the setting **Combine HTML images into image sprites** in the **Images** tab).

Symptom: Acceleration is working but there is a layout problem.

Problem: This is normally style sheet or HTML image related.

Resolution: Reduce the number of style sheet objects and HTML images. This problem typically requires the use of acceleration rules to work around whatever technique is causing this symptom to appear.

CSS errors

You could have image errors if your web pages do not display images in their specified position or display images with incorrect size. Image errors can also be "invisible", as it is the case when Aptimizer creates new resource sets for every new visit to the page.

You can spot the problem with resource set creation by inspecting the elements on the page. You should see the most common elements being created once and served from the cache multiple times.

Symptom: Internet Explorer does not apply background images defined in a stylesheet.

> **Problem:** The cause is the security policy that Internet Explorer is enforcing. If an HTML page on one domain references a resource on a different domain and that resource type is rfc822 then the browser will load it but not process it since it doesn't know if it can be trusted. External resources on the same TLD as the page are implicitly trusted.
>
> **Resolution**:
>
> - Change the CDN hostname to be a sub domain of the main web site so that resources are implicitly trusted.

JavaScript errors

It's important to ensure any Javascript errors, as detected by most modern browsers, do not exist before enabling Aptimizer. This will help determine if these problems are caused by any of the introduced optimizations. If in doubt, request the page again with Aptimizer disabled. This will bypass the accelerator and you can then check if JavaScript errors exist regardless of the optimizations.

On Internet Explorer 9 you have access to Developer Tools and debug information. On Firefox and Chrome you can install the Firebug add-on and access similar information. Most browsers offer an identifying message/notification when JavaScript errors are encountered regardless of installed tools and plug-ins. This message is usually helpful in disclosing which script is causing problems and frequently also identifies the specific position or line number of the erroneous section of code.

Debugging JavaScript errors

Because Aptimizer allows you to apply rules to specific URLs, you can switch off certain optimizations for individual (or a group of) resources without compromising the default settings for your web site.

To resolve JavaSript issues with Aptimizer enabled you can perform one or more of the following:

- Disable the **JavaScript optimization** option on the **JavaScript** tab of any matching rules;

- Identify which script is breaking and create a URL rule to exclude it completely from optimization;

- You can verify Aptimizer is causing JavaScript errors quickly by excluding **all** Javascript optimization on your web site. Create a temporary URL rule to exclude all optimization for any URLs containing ".js".

Common JavaScript errors

Symptoms: Scripts execute with unpredictable results.

Problem: Errors due to an external script reference having been moved in the page.

Resolution: If order-dependent code exists between the external script and other script files or an inline script block, then this may prevent normal functioning of the site. Best practice for building JavaScript libraries is to treat them as libraries of functions, and not execute any immediate code. This allows script to be combined to the maximum extent possible. To work around poorly designed script you can instruct Aptimizer to exclude them entirely via a suitable URL rule.

Symptoms: Scripts fail to load and execute.

Problem: JavaScript frameworks that dynamically load script libraries into the page.

Resolution: These frameworks typically look for a reference to one of the scripts that they use, or will use some sort of registration function to inject the scripts that should be loaded. Since Aptimizer refers to combined scripts using a surrogate URL, this can confuse frameworks that behave like this. The solution in this case is to either exclude these scripts, or transform the loading code such that it understands Aptimizer modifications.

Symptom: Some JavaScript is not executed on a web page when using the JavaScript async option.

Problem: Some ad blocking software will incorrectly filter out the Aptimizer code injected to asynchronously load and execute JavaScript.

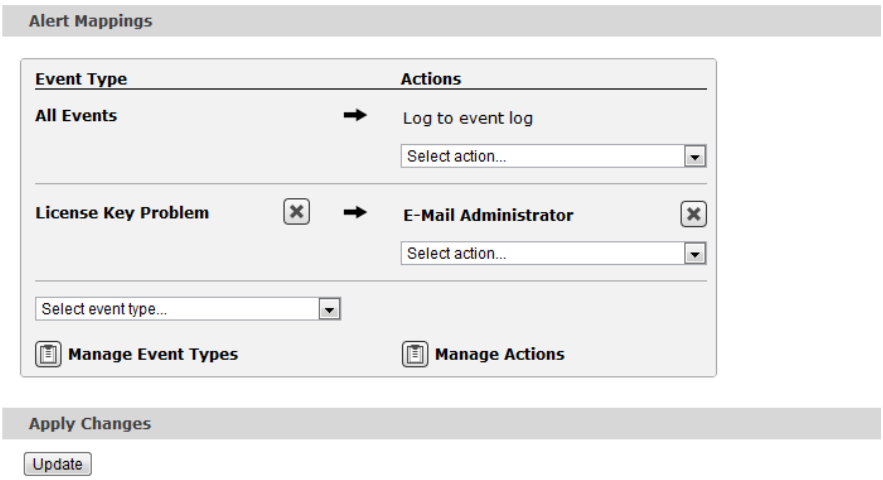
Resolution: Use a URL rule to exclude required scripts from the async code, or disable the option to load and execute JavaScript in asynchronous mode.

CHAPTER 21 Event Handling and Alerts

The event handling capability allows the administrator to configure precisely what actions the traffic manager should take if particular events occur. Event handling behavior can be configured in the **System** section of the Administration Interface, under the **Alerting** tab.

Overview

An Event Handler specifies the actions that should be performed when an event of a particular type occurs. Event Handlers are configured on the **Alerting** page:

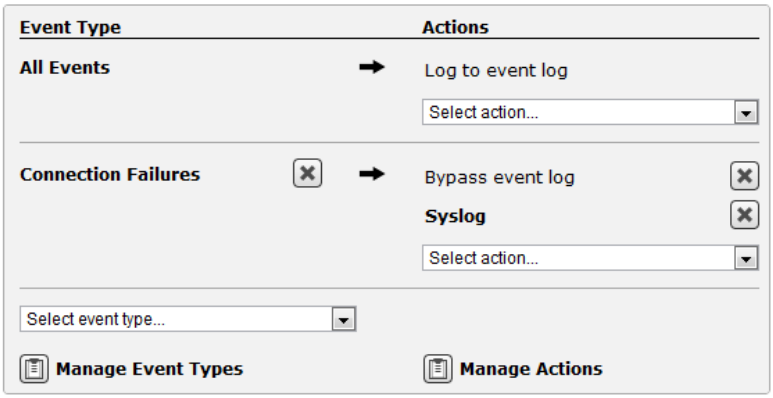


Configuring Event Handlers

The traffic manager contains one built-in event handler that causes all events to be written to the global Event Log. This built-in handler and its "Log all Events" action cannot be deleted, but you can add additional actions if required.

You can add additional event handlers, such as the handler illustrated above that sends an email when there is a problem with a license key. Start by selecting the Event Type from the drop-down box, then select the actions that should be performed when an event of that type occurs.

Note: when you create an additional Event Handler, you can add the internal action 'Bypass event log'. Any events that are processed by that Event Handler will not be logged to the global Event Log:



Events that match the Connection Failures Event Type will be handled by the Syslog action, and will not be logged to the Stingray Event Log.

Event Types

The traffic manager can respond to a wide variety of events, such as node failures, Traffic IP transfers, Service Level failures and custom events raised from a rule. For convenience, individual events are grouped into a number of pre-defined types, and you can create new Event Types as required.

All Events	Includes all of the events that the traffic manager can detect; a built-in handler will log all events that match this type to the global Event Log.
Critical Problem Occurred	Includes all of the events that indicate a serious problem, including internal failures, node failures, IP transfers (because a traffic manager has failed) and other hardware problems.
Critical Problem is Resolved	Includes events that indicate that a problem that was raised in the 'Critical Problem Occurred' Event Type has been resolved; for example, a failed node has started working again. If you configure the traffic manager to send an email message when a problem occurs, it is sensible to also configure the traffic manager to send an email message if the problem is resolved and intervention is

	no longer necessary.
Default Events	Included for compatibility with earlier versions of the traffic manager; it includes all of the events that would have caused the traffic manager to send an email message or raise an SNMP trap. When the traffic manager is upgraded from an earlier version, the upgrade process will add an Event Handler to preserve the earlier behavior if necessary.
Infrastructure Problem	These events are raised if the traffic manager detects a problem with an item of infrastructure; a server becomes un-contactable for example.
Service Problem	These events are raised if a problem is detected that could impact the successful delivery of a service.

Some of the predefined Event Types

You can view the Event Types using the **Manage Event Types** button on the **Alerting** page.

Creating new Event Types

You cannot modify the built-in Event Types, but you can make a copy that can be edited, or you can create new Event Types from scratch.

For example, suppose that you use the traffic manager to manage several services, and you wish to create an Event Type that is matched when a particular service fails. You could then use that Event Type to create an event handler that emails the individual responsible for that particular service.

To do this, create a copy of the 'Service Failed' Event Type, and edit the copy so that it is only matched when particular virtual servers, pools and nodes raise events:

1. Click on Manage Event Types, and click on the 'Service Failed' Event Type.

2. You cannot modify this Event Type, but you can make a copy; copy it to an Event Type called 'Custom Service Failed'.

3. You will be presented with the Edit page for your new Event Type. Events are arranged in a tree structure; you can expand the nodes in the tree to see all of the events that the traffic manager can detect. The particular events that the Event Type matches are opened so that they can be easily seen.

4. Edit the new Event Type. You may want to change the list of health monitors, nodes, pools, rules, virtual servers and other objects that raise events so that the Event Type is only matched when the objects relating to your specific service raise events.

5. Apply your changes, then create an appropriate Event Handler for the 'Custom Service Failed' Event Type.

Actions

When an event handler is triggered, the traffic manager will invoke the actions configured for that handler.

Note: The traffic manager contains three built in actions – **E-Mail**, **SNMP Trap** and **Syslog**. Note that on a new install of the traffic manager, the E-Mail and SNMP Trap actions are not completely configured (you will need to edit them to specify the required settings). However, when the traffic manager is upgraded from an earlier version, these settings are preserved.

You may create additional actions of the following types:

E-Mail	This action sends a report of the event(s) that triggered the handler to the email addresses configured in the action. You can create several E-mail actions with different email addresses, and use this to configure the traffic manager to email different individuals for different Event Types.
Log to external file	Events are always logged to the global event log, but this action can be used to additionally

	log events to a named log file. The format used is the same as the format used in the global event log. This action is useful when creating a dedicated log of particular events. *
Log to syslog	This action will send a message to the syslog daemon on the named server. This action can be used to send particular events to an external monitoring and management tool that can accept syslog messages.
Program	This action runs a script or other program that you have uploaded to the traffic manager system using the **Upload and Manage Programs** button on the action's **Edit** page. Refer to the *Calling a program or script* section of CHAPTER 21 for an example program.
SNMP Notify/Trap	This action sends an SNMP Notify or Trap to the named SNMP receiver application on the host specified. Traps are defined in the Stingray MIB, which you can download from the action's **Edit** page. Stingray supports SNMP v1, v2c and v3. The configurable settings for this action are dependent on the version you specify on the Edit Action page, and directly mirror the available settings on the **System > SNMP** page (SNMP command responder options). Please refer to the *Monitoring Performance using SNMP* section of CHAPTER 7 for more details.
SOAP Callback	This action sends a SOAP message to the named server. Refer to the *Sending a SOAP message* section of CHAPTER 21 for an example of a SOAP action and a server that handles it.

*** Stingray appliance/cloud users: When using this facility on a virtual or cloud instance, you should ensure your file is kept within ZEUSHOME. When applying an upgrade, all user-created files and directories outside of ZEUSHOME are likely to be lost.**

On the **Alerting** page, click the **Manage Actions** link to edit actions and create new actions.

You can also apply the internal Action 'Bypass event log' to prevent events being logged to the global event log.

Testing Actions

When you create or edit an action, you can invoke it immediately with a 'test' event. This is a useful way of testing the correct configuration and operation of your actions.

Configuring an Event Handler

Before you configure an event handler, you will need to create an Event Type that matches the events you would like to trigger the event handler, and create an Action that performs the correct task, such as sending an email.

The event handler links the Event Type with one or more actions that should be invoked when an event that matches the Event Type is raised.

To configure an event handler, go to the **System > Alerting** page. Select the Event Type from the drop-down list to create a new event handler, then select the Actions you wish to be invoked from the drop-down in the entry for that event handler.

Press 'Update' when you have completed your changes.

Duplicate Events

An event may match several Event Types, and therefore trigger several event handlers. If the same action is configured on each of these handlers, it will be invoked multiple times.

To reduce the volume of small emails sent, the E-mail action is not triggered immediately, but will gather email messages and send them at 30 second intervals. You can configure this behavior of the email action in the **System > Global Settings** page, in the **Other Settings** section.

If a minor event is raised several times in quick succession, the traffic manager will suppress the repeated events. Major events are never suppressed in this way.

Custom Actions

Calling a program or script

The Program action is used to run a program or script that you have uploaded to the traffic manager. Programs and scripts are uploaded using the **Catalogs > Extra Files > Action Programs** page.

When this action is invoked, the traffic manager will execute the program in the background, passing it command line arguments that identify the event that triggered the action.

The traffic manager passes the following arguments to the program or script:

- Any command line arguments you have configured in the action (optional);
- An argument named `alertname` that provides the name of the Event Handler that invoked the action that executed the program;
- The event description; this is in exactly the same format as the log line for the event that is written to the Event Log.

For example, if a TrafficScript rule raises an event named TestEvent, and this is caught by an event handler named 'My Event Handler' which invokes a Program action, the program will be called as follows:

```
programname "--alertname=My Event Handler" \
   "INFO trafficscript/TestEvent CustomEvent
rules/<rulename> \
   vservers/<vservername> <description>"
```

Note that the script is given two arguments (linebreaks have been inserted for readability).

Sending a SOAP message

A SOAP action sends a SOAP request to the proxy configured in the action. The SOAP message conforms to the WSDL specification in:

```
ZEUSHOME/zxtm/etc/wsdl/AlertCallback.wsdl
```

You can download this file from the **Actions > Edit** page when you edit a SOAP action. This specification is documented in the Stingray Control API documentation.

The SOAP request uses the interface `http://soap.zeus.com/zxtm/1.0/AlertCallback`, and invokes the method `eventOccurred`, passing it the following arguments:

- The name of the traffic manager machine that raised the event (type `xsd:string`);
- The time of the event, in the `xsd:dataTime` format[11];
- The severity of the event (of the enumeration type `AlertCallback.Severity`);
- The tag that identifies the event (of the enumeration type `AlertCallback.Tag`);
- An array of tags; this is empty and reserved for future use;
- An array of objects (type `AlertCallback.ObjectArray`); these objects (type `AlertCallback.Object`) describe the object that was responsible for raising the event (such as a node or rule).

  Some events will reference multiple objects. For example, an event raised from a TrafficScript rule will contain three objects that identify the event name, the rule and the virtual server running the rule;
- A human-readable description of the event (type `xsd:string`). This is predefined for most events, but when an event is raised by the TrafficScript function `event.emit()`, the description contains the message provided by TrafficScript;
- Additional data provided by the action (type `xsd:string`);
- The name of the Event Handler that invoked the action.

The following Perl CGI script implements a simple SOAP consumer for the AlertCallback messages; this responder logs the details of the message to a local file. You can install this CGI script on a suitable web server (equipped with Perl and the

[11] http://www.w3.org/TR/xmlschema-2/#dateTime

SOAP::Lite Perl modules), then configure a SOAP action, setting the proxy value to the URL of the CGI script.

```
#!/usr/bin/perl

package AlertCallback;

sub _addSerializerForEnum($;$)
{
   my ( $xsdtype, $ns_prefix ) = @_;
   $ns_prefix = "zeusns" unless $ns_prefix;

   $func = "SOAP::Deserializer::as_${xsdtype}";
   *$func = sub { return $_[1] };
}

BEGIN {
   _addSerializerForEnum( "AlertCallback.Tag" );
   _addSerializerForEnum( "AlertCallback.ObjectType" );
   _addSerializerForEnum( "AlertCallback.Severity" );
}


sub eventOccurred($$$$$$$$)
{
   my $self = shift;
   my( $stm, $time, $severity, $tag, $tags, $objects,
      $desc, $additional, $handler ) = @_;

   open LOG, ">>events.log" or die( "Couldn't open log
events.log: $!" );

   print LOG "$stm - $time - $severity: $tag [";
   print LOG join ', ', map { "$_->{type}:$_->{name}" }
@$objects;
   print LOG "] $desc ($additional); Event handler:
$handler\n";

   close LOG;
}

package main;

use SOAP::Transport::HTTP;

SOAP::Transport::HTTP::CGI
 -> dispatch_with(
    { "http://soap.zeus.com/zxtm/1.0/AlertCallback/" =>
'AlertCallback' } )
 -> handle;
```

Perl CGI script to log SOAP action messages

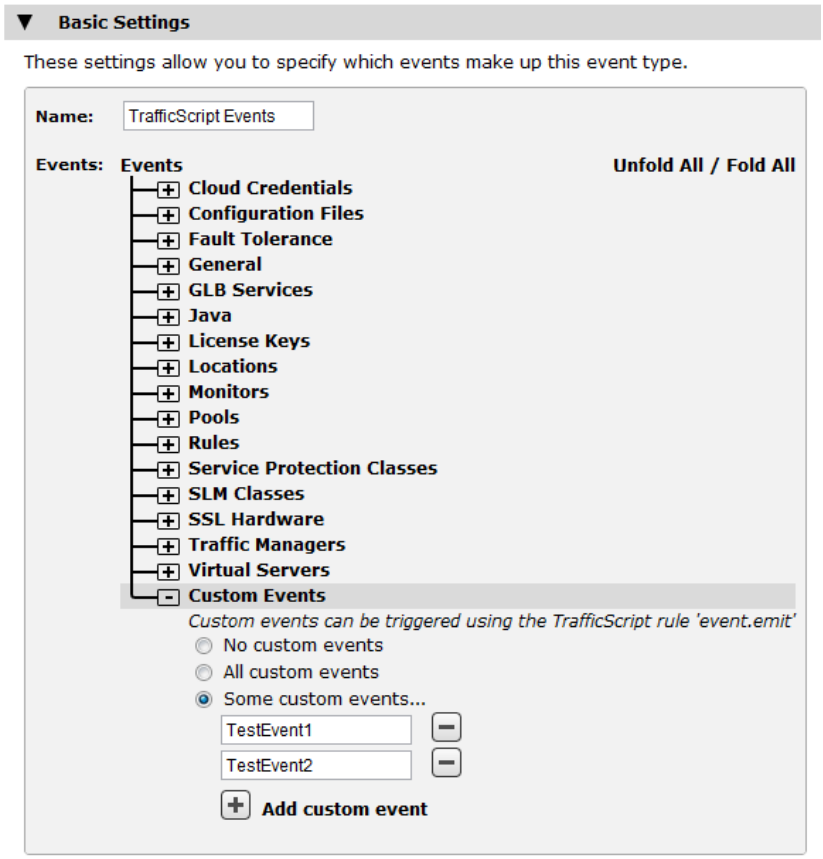
If your action is invoked by multiple different events, you may wish to determine the precise event based on the tag value in the SOAP message. You can do so by matching the description of the event in the event tree (when you configure a custom Event Type) with the description in the WSDL file to determine the value of the `AlertCallback.Tag`.

Raising events from TrafficScript or Java Extensions

You may raise an event from within a TrafficScript rule or a Java Extension. The appropriate event handlers will be run asynchronously, in the background, and the TrafficScript rule or Java Extension will not block.

Events are raised using the `event.emit( event_name, event_message )` function (TrafficScript) or the `emitEvent( event_name, event_message )` function (Java).

To invoke an action based on an event raised using `event.emit()` or `emitEvent()`, you need to create an Event Type that includes Custom Events. You can choose to match all custom events, or just specific events:



An Event Type that matches the TrafficScript events 'TestEvent1' and 'TestEvent3'

The names of the specific events in the Event Type should match the name used when the event was raised in the TrafficScript rule:

```
# raise an event named "TestEvent1", to be caught
by an Event

# Type containing a custom event called
"TestEvent1".


event.emit( "TestEvent1", "The request was ".$url
);
```

Example

A simple intrusion detection system may wish to ban all remote clients that attempt to access a privileged URL:

```
# TrafficScript Rule

$path = http.getPath();


if( string.StartsWith( $path, "/secure/admin" ) )
{

   $ip = request.getRemoteIP();

   if( !string.ipMaskMatch( $ip, "10.0.0.0/8" ) )
{

      event.emit( "UnauthorizedIP", "IP: ".$ip );

   }

}
```

A TrafficScript rule which detects unauthorized accesses and raises an event

You should create an Event Type that contains the custom event named "UnauthorizedIP", and an event handler based on that Event Type.

The event handler could then invoke a custom program action that updates the intrusion detection rules to ban connections from that IP:

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


my $logLine = $ARGV[1] or exit;


$logLine =~ /IP:([\d]+\.[\d]+\.[\d]+\.[\d]+)/;
my $ip = $1;


open OUT, ">>/tmp/events.log" or die $!;
print OUT "The bad IP was $ip\n";
close OUT;


# reconfigure upstream firewall, or Stingray's
banned IP list
# to ban access from $ip
```

An Action script to process event messages raised by the above TrafficScript rule

CHAPTER 22 Configuring Stingray in a virtual or cloud environment

Note: This chapter applies to Stingray virtual appliance and cloud variants. It does not apply to software-only variants.

Stingray virtual appliance and cloud instances contain a number of additional configuration pages to manage other settings such as network configuration, and time and date. On these variants, the Administration interface performs a number of additional system checks and tests, such as checking the integrity of various operating system components and features.

These pages are not available on software-only variants because these settings are configured by a system administrator responsible for the host operating system or virtualization platform.

Please refer to the appropriate **Installation and Getting Started Guide** that was shipped with your Stingray variant for initial setup and configuration documentation, and instructions on how to reset Stingray to its initial factory configuration.

Network Configuration

Your traffic manager instance contains a number of network interfaces, and requires a hostname to identify it. The **Networking** page in the **System** section of the Admin Interface allows you to configure these settings, along with DNS, Routing and IP forwarding and NAT.

Note: Unlike most other settings in the Admin Interface, any settings that you configure on this page only apply to the local Stingray instance whose Admin Interface you are accessing. If you want to inspect or configure the network settings for a different traffic manager in your cluster, you should use the Admin Interface for that machine.

Configuring the Hostname and IP addresses

The hostname uniquely identifies your traffic manager system within your cluster, and the IP addresses identify each active network interface on the system. If you do not use a management port, the hostname you select must resolve to a permanent IP address on the traffic manager system, so you will need to configure your local DNS accordingly.

Typically, the first network interface (eth0) is designated as the management port and the hostname you specify does not need to resolve to the IP address on that machine. The traffic manager will only accept and transmit management information on the management IP, and it should be connected to a secure network, separate from networks that route public internal or external traffic. Additional network interfaces are used to manage external or internal traffic.

If you chose not to use a management port, then the traffic manager will accept management traffic on all interfaces, and the hostname you specify must resolve to one of the IP addresses active on the traffic manager system.

The **Installation and Getting Started Guide** describes which interfaces are present on your traffic manager, and how they are numbered.

Configuring networking

By default, the traffic manager auto-negotiates network settings with the switch each interface is connected to.

It is sometimes necessary to manually specify the interface settings. For example, many switches may fail to negotiate the fastest speed they support. Note that 1Gb speeds are auto-negotiated and cannot be forced.

Configuring your DNS settings

The traffic manager will query a local DNS server for inter-cluster communications, when resolving node names to IP addresses, when communicating with external services and whenever access restrictions or other configuration required it to resolve the IP address of incoming network connections.

You will need to configure your traffic manager system with the IP addresses of one or more local DNS servers. These are normally configured when you first install the traffic manager, but you can edit this configuration and add fixed host-to-ip-address mappings using the **DNS** part of the **Networking** page in the Admin Server.

Configuring additional routes

Your traffic manager will route traffic through the most appropriate network interface according to the IP addresses you have assigned to these interfaces. You should also configure a default gateway to which the traffic manager can forward non-local network traffic.

Important: Your default gateway is used in network connectivity tests, as described in the *Understanding a traffic manager's fault tolerance* section of CHAPTER 6. It should reside on the network used for external (incoming) traffic. Do not specify a gateway that resides on the management network, or configure the **flipper!frontend_check_addrs** with more appropriate IP addresses.

If necessary, you can manually add additional routes to non-local networks when the default gateway is not appropriate. You must specify the non-local network (destination and netmask), the local gateway to that network (optional) and the interface to use to access the non-local network.

These settings are managed in the **Routing** part of the **Networking** configuration page in the Admin Server.

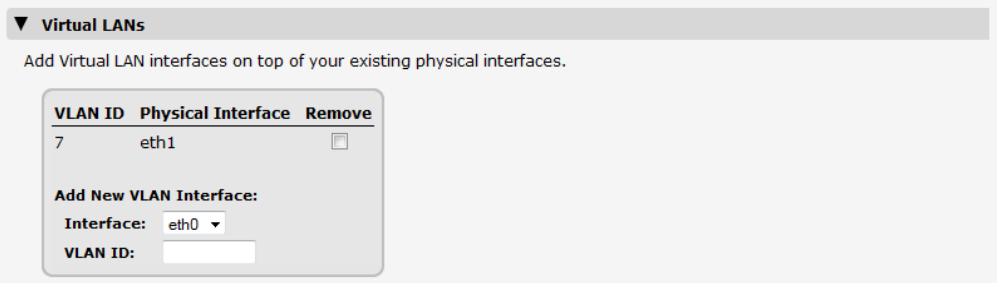
Trunking/Bonding

If you configure the same IP address on multiple physical network interfaces, the traffic manager will make the interfaces members of a trunk (applicable only to virtual appliances). For this to work, your switch must support IEEE 802.3ad, and may need to be reconfigured to enable it. Your switch may refer to this feature as link aggregation, bonding or trunking.

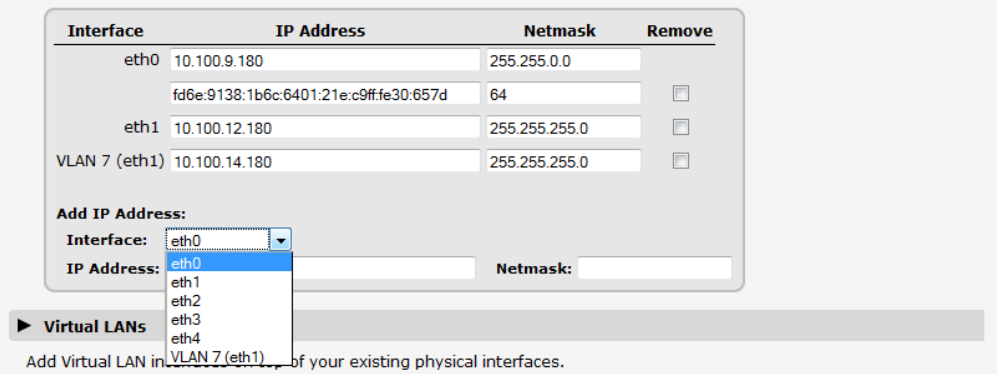
Configuring VLANs

You can use the Stingray Administration Interface to configure VLANs on any of the network interfaces or bonded/trunked interfaces on your traffic manager. This configuration is performed from the **System > Networking** page:

1. Add a Virtual LAN interface to the appropriate physical interface:



2. Configure the networking (IP address and subnet) on the new VLAN:



Any traffic that is routed out through the VLAN interface will be tagged with the VLAN ID.

Configuring Network Address Translation (NAT)

Depending on your network topology, your traffic manager may function as a network gateway for your back-end nodes or other devices on your network. For example, if you use IP Transparency, you need to configure your back-end nodes to use the traffic manager as their default gateway (see the *Routing configuration* section of CHAPTER 2).

When Stingray (or any other network device) acts as a gateway, it forwards traffic from the source nodes to their destination, and forwards responses back to the source nodes.

If all of the traffic that Stingray forwards comes from sources with publicly-routable IP addresses, then there are no complications. All hops in the link between the source and destination will know how to route the responses back to the source IP addresses.

However, it is quite common to locate your back-end nodes on a private network that is not routable from the public internet. This is done to conserve public IP addresses and to add a degree of security because external internet clients cannot send network traffic to these nodes directly. In this case, a technique called NAT is used by a gateway to transparently substitute the un-routable source IP address with the publicly routable IP address of the gateway.

Your downstream router may perform NAT for you, and your internal systems may all know how to route internal traffic through the Stingray instance correctly. If this is not the case, you can configure the traffic manager to perform NAT on forwarded packets, to ensure that the responses are routed back through the traffic manager gateway.

Use the **NAT** section of the **Networking** page in the Admin Server.

Select which interfaces should perform NAT on network traffic that is forwarded on behalf of an internal source.

Typically, these interfaces will have publicly routable IP addresses, and will be located on the external side of your network (for outgoing traffic).

Note that NAT will not work if the private and public IP addresses are configured on the same network interface. You must have at least two network interfaces to create a working NAT configuration.

This has the following effect:

Any traffic that is forwarded out of this interface is NAT'ed: the source IP address in the traffic is replaced with the IP address of the gateway interface (the source port number may also be replaced with an arbitrary port number).

From the destination node's perspective, the traffic appears to originate from the traffic manager's gateway IP address. The destination node (and all intermediate hops) should know how to route traffic back to that IP address.

When the destination node replies, Stingray receives the reply on the NAT interface. The original source IP address and port is then inserted into the reply and forwarded on to the back-end node that originated the connection.

Time and Date configuration

Note: The settings described in this section will vary slightly for Xen virtual appliance users. Time is synchronized with the host hardware, so the only modifiable setting in this case is the timezone.

It is useful to ensure that the clocks on your traffic manager and your back-end nodes are synchronized with each other, so that when you compare log files from each machine, you can determine how events that occurred on each system are related.

If the times are not in sync with each other, it is harder to relate events to each other, such as whether an error logged by the traffic manager corresponds to an error recorded by a back-end server.

Use the Time page in the System section of the Admin Interface to manage the time settings on the local Stingray instance.

Note: Unlike most other settings in the Admin Interface, any settings that you configure on this page only apply to the local Stingray whose Admin Interface you are accessing. If you want to inspect or configure the time settings for a different traffic manager machine in your cluster, you should use the Admin Interface for that machine.

Setting the Time manually

Use the Basic Settings section to view the current system time, date and timezone, and to set the settings manually if required.

Using an NTP server

If you configure all your Stingray instances and your back-end servers to use the same NTP server, this will ensure that their clocks are set accurately and are closely synchronized with each other.

You can use public NTP servers such as pool.ntp.org, but due to network latency or outages, these may not be sufficiently reliable. You are recommended to run a local NTP service and synchronize all of your local systems to that server. Your local NTP server could itself synchronize from an NTP server managed by your ISP, or a public NTP server.

If for some reason, the time on your Stingray virtual appliance/cloud instance differs from the correct time by a significant amount e.g. 30 minutes, then NTP will not adjust for such a large difference. To correct the time difference in this case, you can go to the **Time** page in the **System** part of the Admin Server and click on "Sync Time Now". Your time should now be correct.

Synchronizing time from Stingray

Because it typically spans multiple networks, a Stingray appliance/instance may be a suitable, consistent local time source for a variety of networked devices. Stingray runs a local NTP server that is accessible on all interfaces, so you may wish to configure your other servers to synchronize time from there.

Remote Login to Stingray

Should you ever need to access the Stingray virtual appliance/cloud instance's command line, you may log in using SSH, and the username and password of a user in the 'admin' group, or a different group with 'Appliance Console' permissions (see the *Permission Groups* section of CHAPTER 24).

Users who log in with appropriate credentials will obtain root access to the appliance/instance; their home directory is `/root`.

If you wish to configure password-less logins using SSH public/private key identification, the SSH configuration directory is `/root/.ssh`.

Cloud Steelhead Integration

Note: This section applies only to *VMWare ESX* and *Amazon EC2* Stingray variants. Please refer to your support provider for more details.

Stingray Traffic Manager can be configured to integrate with Riverbed's Cloud Steelhead appliance in order to provide optimized web services and traffic flow to and from your cloud computing or data-center infrastructure.

A client trying to access web applications hosted in a cloud environment would normally attempt a connection to one of the traffic IPs configured on your Stingray instance, which would then perform all usual traffic routing and manipulation according to your traffic management policy for that service. This connection is then usually passed to one of the free back-end nodes that host the service being requested.

A Steelhead can incorporate WAN optimization into this arrangement, although you will require a Steelhead appliance at each end of the connection to facilitate this. Given the circumstances where you are able to route your traffic through a regular physical or virtual Steelhead appliance at the remote client location, you will also need to configure Stingray to acknowledge receipt of the optimized traffic and in turn contact a Steelhead in the cloud/data-center to complete the connection.

This Cloud Steelhead is effectively inserted into the traffic flow as a proxy to maintain an optimized connection to the client-side Steelhead, which in turn passes traffic to Stingray in order to balance services to the back-end server nodes as usual.

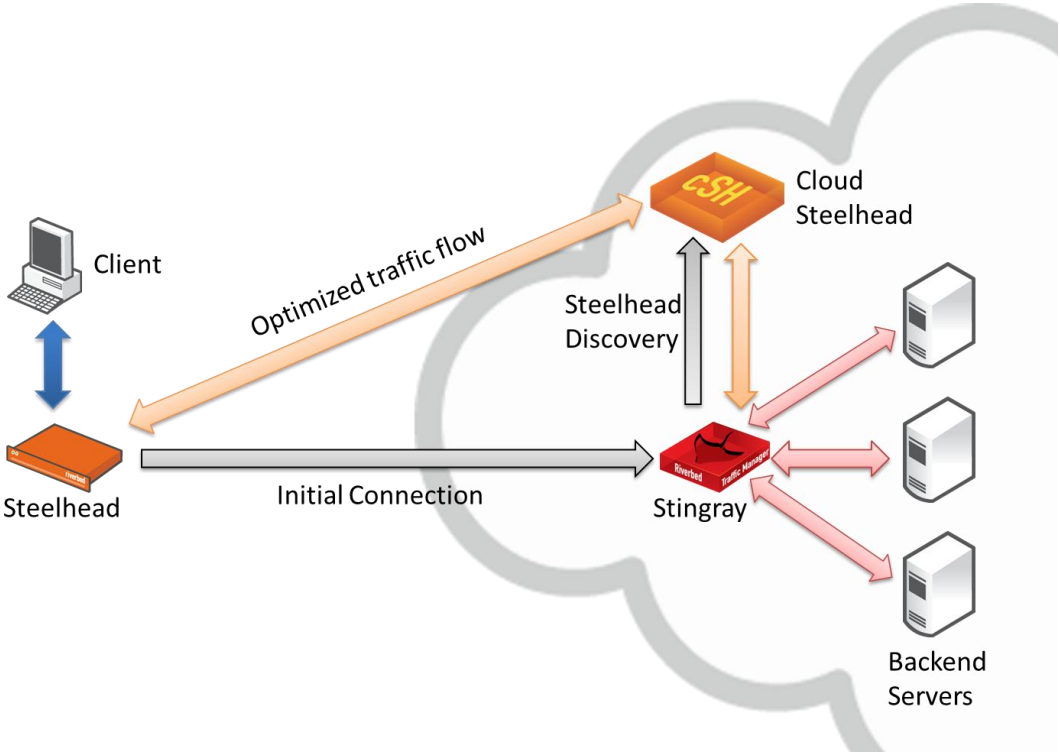


Fig. 40. Optimizing the traffic flow to your cloud-based web services

Stingray provides a feature called the *Cloud Steelhead Discovery Agent* that can be used to intercept any traffic identified as optimized and forward it to a Cloud Steelhead in the local data-center or cloud infrastructure. The data is then returned to the traffic manager for processing in the normal way. Hence the Cloud Steelhead is inserted into the regular traffic flow between the client-side Steelhead and the traffic manager.

The Discovery Agent is controlled from the **System > Steelhead** page of the Admin UI. On this page you can provide settings that enable the discovery agent and configure which Cloud Steelheads this traffic manager should use to optimize traffic.

Important: Steelhead Discovery Agent settings are configured on an **individual machine basis**. If you are joining one or more traffic managers together to form a cluster, you should ensure that each machine has these settings provided as no automatic config replication will be performed for this function.

The available config keys are as follows:

`Enable`	Setting enabled to **Yes** starts the agent and attempts to connect to a Steelhead using the configured discovery mode.
`Mode`	The mode used to discover your Cloud Steelheads in the local data-center or cloud. It can be one of:

- **Riverbed Portal**

  Use the standard Riverbed cloud portal to manage your Cloud Steelheads. This is usually only used on Amazon EC2.

  You must also configure the following settings when using this mode:

Client ID	The identifier for this server listed on the Riverbed cloud portal.
Client Key	The key for this server listed on the Riverbed cloud portal.
Proxy Hostname/IP	If the Riverbed cloud portal is accessed via a proxy, enter its hostname or IP address. Otherwise leave it blank.
Proxy Port	If a proxy hostname or IP address is specified you must also enter the port to use.

- **Local Portal**

  Use an alternate portal to manage your Cloud Steelheads.

  You must also configure the following settings when using this mode:

Portal Hostname/IP	The hostname or IP address of the portal you are using to manage your Cloud Steelheads and servers.
Client ID	The identifier for this server listed on the portal.
Client Key	The key for this server listed on the portal.
Proxy Hostname/IP	If the portal is accessed via a proxy, enter its hostname or IP address. Otherwise leave it blank.
Proxy Port	If a proxy hostname or IP address is

	specified you must also enter the port to use.

- **Manual**

  Manually configure the IPs of your Cloud Steelheads.

  You must also configure the following settings when using this mode:

Steelhead IPs	The IP addresses of all the Cloud Steelheads you want to use to optimize traffic. If using priority load balancing this list should be in order of priority, with the highest priority Steelhead last.
Load Balancing Method	When multiple Cloud Steelheads are available one is selected using either 'Round Robin' or 'Priority' load balancing. In round robin mode each contactable Steelhead is used in turn. In priority mode the last contactable Steelhead in the IP list is used, continuing up the list.

Log Level	This setting controls how verbose the logs produced by the Cloud Steelhead discovery agent are. The level selected is the lowest severity that will be logged.

Additionally, the **System > Steelhead** page provides a Discovery Agent *restart* button should you need to fix an incorrectly performing agent. It also provides a display/download section for recent Discovery Agent logs; the last 30 lines of which are shown (if any are available). These logs can be used to analyse and debug any scenarios where the agent is not performing as you expected.

CHAPTER 23 System Security

This chapter covers important aspects of running your traffic manager software in a secure environment. The points in this chapter that consider operating system configuration only apply to the software version, not the virtual appliance or cloud variants.

Firewall and Operating System settings

The traffic manager is not a network-level firewall and it is highly recommended that any traffic manager system is placed behind a firewall of your choosing. There are extra measures that can be taken to reduce potential security problems related to your choice of operating system, and UNIX security in general. These are discussed briefly below.

Important: The notes that follow are brief and deal specifically with Stingray traffic management products. Customers are advised to take extra care over all security considerations related to Internet service providing, and to seek specialized advice if required.

Firewalling Techniques

The traffic manager must be able to respond to requests, and certain protocols (such as FTP) may need to open additional connections with clients. Stateful firewalls can be configured to allow outgoing connections to be established from the traffic manager on any port, and for subsequent responses from clients to be allowed back to the server.

The recommended firewall configuration for the traffic manager is therefore a stateful firewall that denies all inbound traffic by default, allowing named ports on the traffic manager to be contacted and any responses to the traffic manager to be allowed through the firewall.

Firewall configuration with Stingray traffic management

The traffic manager requires some ports and protocols *in addition* to those used by the virtual servers configured. These are:

The Admin Server port (HTTPS). This is usually port **9090**, and is the port you use, with the machine name, to browse to the Admin Server interface. This port must be open to any machine from which you wish to access the Admin Server interface.

The control port. This defaults to **9080** and is used by the traffic manager control protocol to communicate changes in configuration data between traffic managers. This port does not need to be exposed outside the cluster.

> You can check the control port number in *ZEUSHOME*/zxtm/global.cfg.

The traffic manager also uses broadcasts to identify other traffic manager servers. These broadcasts are sent to the multicast address, **239.100.1.1**, port **9090**; all of the traffic managers in a cluster listen on this address. This IP address and port can be adjusted from the **Traffic Manager Fault Tolerance** settings under the **Global Settings** tab.

If you have configured each traffic manager in your cluster to use a dedicated Management Network as described in the *Dedicated Management Network* section of CHAPTER 2, this management traffic will be restricted to that network alone. It should not be possible to connect to any of the traffic manager management ports from another network. Nevertheless, you should still firewall these ports off from untrusted sources, for example, in the event that a configuration error relaxes the traffic manager's management restrictions, or the management network is accidentally exposed.

Stingray virtual/cloud instances run a local NTP service that listens for NTP (time) requests on all interfaces (see the *Synchronizing time from Stingray* section of CHAPTER 22). The NTP services runs on port 123 (UDP and TCP). It is safe to firewall these ports off if you do not wish to use Stingray as a local time source.

Important: One of the tests the traffic manager uses to detect network availability is ping. The traffic manager may not function correctly if your firewall or local TCP/IP tunings disable or rate-limit **ICMP** packets to the default gateway, the back-end nodes and the other traffic managers in the cluster.

Network Design

The traffic manager is a proxy with advanced traffic management capabilities. This means that the traffic manager is normally placed between two networks, relaying requests from a public network to a private network. Each traffic manager must be able to contact machines on both the traffic and back-end networks.

For example, the back-end pools may be on an RFC1597 private network, with IP addresses in any of these groups:

```
10.0.0.0/8
172.16.0.0/12
192.168.0.0/16
```

These networks are not routable on the Internet. They offer both convenience in terms of allocating internal network space, and security, since a correctly configured external router will not accept packets for these networks.

It is therefore most secure for the traffic managers to have traffic IP addresses (traffic IP groups) that are connected to the Internet, and for all back-end pools to use RFC1597 network addresses on a separate physical network (or correctly configured VLAN). This then prevents back-end servers from being reached directly from the Internet, and allows the traffic manager to manage all inbound connections securely and efficiently.

Furthermore, you may choose to use a separate RFC1597 private network for internal management traffic, including access to the Stingray Admin Server. See the *Dedicated Management Network* section of CHAPTER 2 for more details.

Unix User Permissions

Stingray traffic management software must be started as root. This allows the `zeus.zxtm` process to bind to privileged ports (those below 1024). For example, port 80 (HTTP) is a privileged port. The traffic manager also uses superuser privileges to provide fault tolerance; these privileges are needed to raise and lower Ethernet interfaces.

The traffic manager processes that manage incoming traffic (i.e. untrusted input) relinquish superuser privileges once they have started. This part of the software runs with the UNIX user and group specified during installation. The Stingray Admin Server (`zeus.admin`) does use superuser privileges to manage configuration that is owned by the root user.

For details of the installation process please see the **Installation and Getting Started Guide**. If you are uncertain of the permissions of the traffic manager process, you can check which user owns the `zeus.zxtm` processes using `ps`, or by inspecting *`ZEUSHOME`*`/zxtm/global.cfg`. The user is usually `nobody`, and the group `nobody` or `nogroup`.

If you wish to change the permissions, the recommended way to do this is to run the traffic manager 'configure' script at *`ZEUSHOME`*`/zxtm/configure`.

You must do this as the superuser. For more details please see the **Installation and Getting Started Guide.**

File System Security

All files within *ZEUSHOME* are configured during installation, and subsequent alterations are made by the Stingray Admin Server. All files within *ZEUSHOME* are therefore owned by the superuser. Certain files have world-readable permissions flags, in order to allow the `zeus.zxtm` processes running with unprivileged permissions to read the traffic manager configuration files.

If required, the traffic manager can be run within a `chroot` jail, effectively preventing all access to the file system other than *ZEUSHOME* itself. You will need to provide the required system libraries, some device entries in `/dev` and some other system-specific configuration information. For further details, see your system documentation.

Note: Syslog logging is based on UNIX sockets and may still write files outside *ZEUSHOME*.

Operating System Settings

For best security, traffic managers should not be used for running applications other than the traffic manager, especially applications that provide network services. When a new traffic manager is commissioned, pre-installation checks should include removing or disabling UNIX services which will not be used. For example, services such as `lpd` (the UNIX print server) and the RPC Portmapper are often started by default, and are not necessary for the traffic manager to function. By uninstalling these services completely, you can dramatically reduce the exposed interfaces that provide a means of accessing your traffic manager servers.

Some services may still be required (e.g. Syslog, secure shell). Where practical, these services should be bound only to the private network interface or to the loopback interface if they are not required externally. This will help avoid attracting unnecessary attention to your servers, and should be done even where firewalls are in place.

A good pre-installation starting point is to have only port 22 open on the server, for inbound `ssh` connections, to the management port only (if configured). Open ports can be checked using the freely available `netstat` and `nmap` tools.

Administration of a UNIX server requires regular operating system maintenance. Security is an ongoing process. In particular, it is best practice to track vendor

patches and to upgrade services that remain exposed to the Internet as soon as new versions with security-related problems are identified. This applies not only to the services that the traffic manager is managing, but also to tools installed on traffic managers, such as SSH servers used for administration. For most operating systems, including Linux, the kernel itself may require upgrades from time to time as security problems are identified and fixed.

CHAPTER 24 Admin Server Security

This chapter describes how to control access to the administration components of the traffic manager system – the web-based administration server, the Control API (and, by implication, the CLI) and remote access via SSH (virtual appliance/cloud instances only).

Basic Administration Server settings

To manage basic security settings, click on the **System** button and then the **Security** tab. This will take you to the **Administration Security** page, where you can configure the SSL certificate, IP-based access controls and the ports used by the Administration Server.

Changing the Admin Server SSL Certificate

The first option, **SSL Certificate**, allows you to change the SSL certificate used for the Stingray Admin Server interface. You can choose to generate a new, self-signed certificate, or upload new certificate and private key files. Click **Update** when you have finished.

Note: You cannot choose a certificate from the catalog, since the Admin Server is not a public service but a separate, external process in the traffic manager framework.

The SHA-1 fingerprint of the SSL certificate is displayed here. This will be useful for the following:

- To verify the SSL certificate when connecting with a web browser for the first time
- To verify the authenticity of traffic manager identities when joining a cluster

It is recommended to make a note of the fingerprint upon setting up a new traffic manager for the first time. For software variants, it will be displayed on the command line after completion of the installation process. For virtual appliances/cloud instances, you can find it on the console.

Should you need to view the fingerprint again, you can display it from the command line using the following:

```
$ZEUSHOME/admin/bin/cert -f fingerprint -in $ZEUSHOME/admin/etc/admin.public
```

Restricting Access to the Admin Server

As well as using the optional management network, the traffic manager can restrict access to the Admin Server using three techniques:

By client IP address (`xx.xx.xx.xx`)

By client IP and netmask (`xx.xx.xx.xx/nn` or `xx.xx.xx.xx/xx.xx.xx.xx`)

By client DNS name or DNS wildcard (`admin.mysite.com` or `*.mysite.com`)

For example, you could specify 10.1.1.1, 10.0.0.0/24 or `*.mysite.com`.

Note: For virtual appliance/cloud instance users, Stingray supports CIDR networks with a netmask of /8, /16, /24 or /32 only.

The traffic manager must perform DNS lookups if you use DNS rules. You must ensure that the necessary DNS settings have been made to allow these lookups to take place.

Once you have clicked **Update**, the traffic manager will check all connections to the Admin Server before allowing access to the login box.

Important: In order for these restrictions to be effective, external firewalls and services must be trusted. IP addresses can be spoofed, and if your DNS service is subverted or hacked then relying on DNS tests may be ineffective.

Changing Admin Server Ports

The Stingray Admin Server is normally run on port 9090 on each traffic manager. You can change this port by adjusting the value in the **Admin Server Port** box. Changes take place immediately when you click **Update**. If you change this port, the traffic manager will redirect you automatically to the new URL.

This will affect any applications that use the Control API, and the connections made by the CLI.

You can specify a dedicated management port (network interface) at installation time. This value cannot be changed from the user interface. To change this value, or to disable the management port entirely, you need to re-run the configure script used in the initial installation.

Please refer to the appropriate **Installation and Getting Started Guide** for details.

Access to the Stingray SSH server

Note: This section does not apply to software-only installations and can be ignored where this is the case.

The Stingray virtual appliance/cloud instance provides command line access through a standard SSH server. This is reserved primarily for log retrieval and certain system maintenance operations and should not be necessary for normal administrative functions.

Restricting SSH access may form part of an organizational security policy. In this case, you can choose to enable or disable SSH access for all users using the `appliance!ssh!enabled` config key. Where access is enabled, you can set the preferred port (usually 22) using `appliance!ssh!port`.

Cluster communication

Communication between cluster members is handled via secure communications designed to withstand man-in-the-middle and other types of attacks or interventions. Stingray provides secure public/private key cryptography using SSL to ensure that your cluster members, regardless of their location in the world, remain able to communicate securely and authentically.

This section allows you to place restrictions on the inter-cluster communications in order to provide the precise level of security for your network setup.

The host IP addresses that can contact the internal administration port on each traffic manager can be specified by the setting `controlallow`. 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.

From version 7.0, each traffic manager has the concept of being either 'trusted' or 'untrusted'. In the case of clusters that span beyond a single trusted network, it is not always possible to know if a cluster member outside this trusted network has been compromised. In order to prevent other traffic managers from becoming compromised, a system has been introduced to enable the administrator to mark certain cluster members as being less secure than others. In other words, if a cluster member is exposed to a higher risk of compromise then it can be marked as such. An untrusted traffic manager can still receive configuration updates, and will still broadcast state/statistical data, but effectively becomes unable to replicate configuration updates out to the other cluster members.

To enable this functionality, each traffic manager in the cluster has a `control!canupdate` flag. This can either be **enabled** for trusted traffic managers, or **disabled** for untrusted ones. You must have more than one traffic manager in your cluster to use this setting.

Note: The Admin UI and Control API on untrusted traffic managers will be disabled for security purposes. If you need to modify machine-specific settings (e.g. Networking, Time/Date, SNMP, or EC2-specific settings) you must do so by first enabling `control!canupdate`. Depending on your network security requirements, it may also be necessary to temporarily remove that traffic manager from the cluster in order to make the change.

A default setting `control!canupdate!default` is provided for new traffic managers to inherit when they join the cluster. You may wish to modify this to **No** prior to joining a new traffic manager from a less trusted location (such as in cloud environments).

Important: As the Admin UI is disabled on untrusted traffic managers, you cannot log on to or make changes to those cluster members. Should you run into a situation where your trusted cluster members become uncontactable for some reason, there exists the risk that you may not be able to administer your entire cluster. In addition, you will be unable to join new traffic managers in order to regain control of the cluster. In order to circumvent this risk, it is strongly advised that you maintain at least one redundant trusted traffic manager in your cluster at all times. Please contact your support provider if you require further assistance.

Important: A compromised pre-7.0 traffic manager *may* remain compromised after the upgrade to version 7.0, regardless of the trusted status given to it. This is because Stingray cannot determine what changes may have been made to circumvent system security prior to the upgrade. It is strongly recommended that you satisfy yourself that a traffic manager in an unsecure location has not been compromised prior to upgrading it.

For the same reason, it is also recommended that you do not mix 7.0/pre-7.0 versions of the Stingray software within a cluster spanning an untrusted network. The inter-cluster security improvements in version 7.0 will not become active until all traffic managers in the cluster are running the same version.

SSL Settings for Admin Server and Internal Connections

These settings control advanced SSL options for connections to the admin server and secure connections internal to the traffic manager. The default settings offer a broad level of security and compatibility suitable for most installations. Consult your system administrator or support provider should you wish to enable compatibility with a specific protocol or connection setting.

User Management

The Admin Server, Control API and CLI each require a username and password to authenticate each connection. This login authenticates the user and, by way of Permissions Groups, defines the authorization the use has to read, write and otherwise manage configuration and system operation.

In order to configure users and groups, click on the **System** button and then the **Users** tab. You will see the **User Management** page.

User Authentication

The Administration Server verifies a user's credentials (username and password) against two authentication sources:

- **Local Users:** Credentials are stored in the administration server (as part of the traffic manager configuration).

  The Administration Server includes a factory-default local user 'admin' that has full administrative permissions (is a member of the 'admin' Permissions Group). In simple environments, users will log in with this account to perform configuration tasks.
  Users cannot be renamed, but you can remove the 'admin' user. You must ensure that another user account with equivalent admin permissions exists so that you can log in and manage your cluster.
- **Remotely authenticated users:** Credentials are authenticated against externally-located systems based on RADIUS, LDAP or TACACS+ services.

  You can configure one or more Authenticators in the User Management page. Authenticators define how the Administration Server verifies usernames and passwords against the database and how it determines the Permissions Group that the user is a member of.

When a user attempts to log in, the Administration Server will first compare the credentials to the list of Local Users and will determine the Permissions Group that the user is a member of. If a match is not found, the Administration Server will try each Authenticator in turn until the user is authenticated and a Permissions Groups is determined.

Local Users

To add a new local user, provide a username and password in the **Create New User** box, and select a group from the drop-down box. This specifies which Permission Group this user should belong to.

To modify the password, Permission Group or UI Preferences of an existing user, click on the username on the **Local Users** page.

UI Preferences for Local Users

Local Users have access to several preference settings that control aspects of the Administration Server UI:

- **use_applet**: enables or disables the status applet that is displayed in every UI page. You may wish to disable this applet to reduce bandwidth, or to reduce HTTP requests when using a tool such as LiveHTTPHeaders or FireBug to monitor HTTP traffic to the traffic manager IP
- **appletwidth**: changes the width of the applet, allowing it to display more (or fewer) bars in the chart that graphs traffic per virtual server
- **trafficscript_editor**: enables or disables the advanced TrafficScript editor, replacing it with a simple textbox when disabled.

To delete a user, click the **Delete User** button at the bottom of the page.

Password Policy

The Password Policy page allows you to configure the password policies applied when local users are created or when they change their passwords. Restrictions on the length of passwords, what character types they must contain, how often they can be changed and how often they must be renewed can be configured here.

Three main security settings are provided:

- **No restrictions:** All passwords are allowed, no restrictions are applied.
- **Default restrictions:** Standard password security is applied to new passwords, specifically:
    - Passwords must be at least 8 characters in length.
    - Passwords must contain at least two alphabetic characters.
    - Passwords must contain at least one uppercase character.

- Passwords must contain at least one numeric character.
- Passwords must contain at least one special, non-alphanumeric character.
- Passwords must not contain repeated consecutive characters, such as 'aaaaa'.

- **Custom restrictions:** The minimum length of passwords and the types of characters they can contain can be specified manually.

In addition, Stingray will maintain a history of passwords used by each user. The setting `password_reuse_after` allows you to specify after how many changes a user can re-use a previous password. This helps to ensure that your users do not simply reset to the same password each time a change is made or required. A value of 0 means a user can re-use any passwords they have previously used.

The setting `password_changes_per_day` specifies how many times a user is allowed to change their password in a 24-hour period. If it is set to 0 then there is no limit to the number of times a password can be changed in one day.

Authenticators

If a user cannot be found in the list of Local Users, the Administration Server will test the credentials against any Authenticators that have been fully configured (and are not disabled). An Authenticator queries an external database, and returns the name of the Permissions Group for users who have valid credentials. Generally the name of the Permissions Group is stored in a field in the remote authentication database, and users who do not have a valid Permissions Group are not allowed access to the Administration Server.

The Administration Server supports LDAP, RADIUS and TACACS+ authentication databases.

Creating an Authenticator

To add a new authenticator, navigate to the **Users > Authenticators** page. Provide a name and select the appropriate authenticator type, then click **Create Authenticator**. This will create an unconfigured authenticator; you will need to provide the appropriate configuration settings for the type you have chosen. Once the authenticator is configured and you have tested it, you can enable it using the 'enabled' setting.

The Administration Server will not use an Authenticator until it has been enabled in this way. If several Authenticators are enabled, the Administration Server will try

each of them (in lexicographic order) until one successfully retrieves a Permissions Group.

LDAP Authenticators

LDAP authenticators have the following configurable settings:

ldap!server	This is the IP address or hostname of the LDAP server.
ldap!port	This is the port used to connect to the LDAP server.
ldap!timeout	The timeout period (in seconds) for a connection to the LDAP server.
ldap!basedn	The base DN (Distinguished Name) for directory searches.
ldap!filter	A filter that uniquely identifies a user located under the base DN. The string "%u" will be substituted with the username. Examples: "sAMAccountName=%u" (Active Directory) or "uid=%u" (Unix LDAP).
ldap!dnmethod	The bind DN for a user can be constructed from a known string ('Construct') or can be searched for in the directory ('Search' - necessary if you have users under different directory paths).
ldap!binddn	Template to construct the binddn from the username, the string "%u" will be replaced by the username. Examples: "%u@mycompany.local" or "cn=%u, dn=mycompany, dn=local". Only used is the dnmethod is 'Construct'.
ldap!searchdn / ldap!searchpass	The bind DN and password 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. These settings are used if the dnmethod is 'Search'.

ldap!groupfilter If the user record returned by **ldap!filter** above 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))`

ldap!groupattr The LDAP attribute that gives a user's group. If multiple values are returned by the LDAP server the first valid one will be used.

ldap!groupfield The sub-field of the group attribute that gives a user's group. For example, if groupattr is "memberOf" which gives something like "CN=mygroup, OU=groups, OU=users, DC=mycompany, DC=local" you would set groupfield to "CN" (the first matching field will be used).

ldap!fallbackgroup If groupattr is not defined above, or is not set for the user, the group named here will be used. If not specified, users with no attribute matching groupattr will be denied access.

RADIUS Authenticators

RADIUS authenticators have the following configurable settings:

radius!server This is the IP address or hostname of the RADIUS server.

radius!port This is the port used to connect to the RADIUS server.

radius!timeout The timeout period (in seconds) for a connection to the RADIUS server.

radius!secret This is the secret key shared with the RADIUS server.

radius!groupvendor	The RADIUS identifier for the vendor of the RADIUS attribute that specifies an account's group. Leave blank if using a standard attribute * such as Filter-Id.
radius!groupattr	The RADIUS identifier for the attribute that specifies an account's group. May be left blank if **radius!fallbackgroup** is specified.
radius!fallbackgroup	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.
radius!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.
radius!nas-identifier	This value is sent to the RADIUS server.

TACACS+ Authenticators

TACACS+ authenticators have the following configurable settings:

tacacsplus!server	The IP or hostname of the TACACS+ server.
tacacsplus!port	The port to connect to the TACACS+ server on.
tacacsplus!timeout	The timeout period (in seconds) for a connection to the TACACS+ server.
tacacsplus!secret	The secret key shared with the TACACS+ server.
tacacsplus!authtype	The authentication type to use. This can be **PAP** or **ACSII**.
tacacsplus!groupsvc	The TACACS+ "service" that provides each user's group field.
tacacsplus!groupfield	The TACACS+ "service" field that provides each user's group.
tacacsplus!fallbackgroup	If **tacacsplus!groupsvc** is not defined, 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.

Testing an Authenticator

The Edit page lets you test an Authenticator, using the **Test Authenticator** section. You can provide the username and password you wish to test; the Authenticator will run and, if successful, return the name of the Permissions Group that the user is a member of. The Authenticator will also provide detailed debugging information to help you fine-tune or correct your configuration if necessary.

Worked Example: Authenticating against Active Directory

You can authenticate users against an Active Directory database using the LDAP Authenticator:

ldap!server	`dir.company.com`
ldap!port	`389`
ldap!basedn	`OU=Company Users, DC=company, DC=local`
ldap!filter	`sAMAccountName=%u`
ldap!dnmethod	`Construct`
ldap!binddn	`%u@company.local`
ldap!groupattr	`memberOf`
ldap!groupfield	`CN`

Test this Authenticator with a username and password:

```
Created LDAP connection to dc2:389
Constructed LDAP user filter:
sAMAccountName=username
Using dnmethod: construct
Constructed user binddn: username@company.local
Attempting to bind using supplied password:
<hidden> (length=9)
Searching for user, parameters: filter:
sAMAccountName=username, basedn: OU=Company
Users,DC=company,DC=local
Search returned 1 matches
Extracted groups using attribute memberOf
   CN=users,OU=Email Lists,OU=Company
Users,DC=company,DC=local
   CN=admin,OU=Email Lists,OU=Company
```

```
Users,DC=company,DC=local
   CN=dev,OU=Email Lists,OU=Company
Users,DC=company,DC=local
Extracting group using field: CN
Extracted groups:
   users
   admin
   dev
No ldap!fallbackgroup defined
Groups returned by authenticator: users, admin, dev
SUCCEEDED, group: admin
```

Permission Groups

Permission Groups are used to assign access to different parts of the Stingray Admin Server to different users. All local and remote users are members of groups; these groups define the access users have to the different aspects of the Admin Server. A number of pre-defined groups are provided with the traffic manager.

To add a new group, provide a name in the **Group name** box and click **Update**. Once the change has been committed, click on the new group name.

Each group has a basic **Description** and a **Login Timeout** setting. The Description can be used to provide a brief explanation of the purpose of this group, and the Login Timeout is the period after which inactive Admin Server sessions will be terminated. All users that belong to this group inherit this timeout value.

Editing the permissions of users in the group

The traffic manager displays a long list of different pages which are available within the Admin Server. All actions are permitted within the ***admin*** group. Your new group starts with all permissions set to **None**.

Each page can have one of the following permissions set for the group in question:

- **None** – group members cannot use this feature.
- **Read Only** – group members can view, but not alter items associated with this page.
- **Full** – members can view and alter items associated with this page.

For example, one of the pages is **Catalog > SSL Certificates**. If you set the permissions to **None**, members of this group will not be able to view the SSL Certificates Catalog page. **Read Only** permissions allow members to view the page

but not change settings on it. If you set the permissions to **Full** then no restrictions apply.

Note: On some pages (such as the SSL Certificates Catalog page mentioned above) there are no configurable settings. Blocking access to a page does not restrict access to those hierarchically below it, nor the settings on them.

Important: Groups with access to the **Users** pages can edit their own account details. You should consider carefully who has these privileges.

Login Timeout

The login timeout is measured in minutes, and is the period of inactivity allowed before the traffic manager ends your session with the Admin Server. Your session is also closed automatically if you close your browser window, or if you log out manually by clicking the **Logout** button.

The login timeout is a property of the Permissions Group. To adjust the login timeout, change the value in the **Inactivity Timeout** box and click **Update**. This setting can be found by clicking on the **System** button, then the **Users** tab, then **Permissions Groups**.

Suspended Users

This page allows you to re-enable one or more users that have been suspended. Users may have been suspended due to exceeding the maximum allowed login attempts, or perhaps for some other administrative reason.

Check the box next to each user you wish to re-enable, and click *Enable Selected Users*.

Note that individual users can also be re-enabled from their **Edit** page, by setting `status` to 'Active'.

Login security and behavior

Password policy can be defined on the **System > Users > Local Users > Password Policy** page to control and place restrictions on user login passwords (see the *Local Users* section of CHAPTER 24). In addition to this, you can define further login behavior settings to provide a greater degree of control and security awareness for the users of your system. The **Login and Security Settings** section of the **System > Global Settings** page provides a number of configuration settings broadly split into two themes:

UI screen banners

Should you need to provide your users with a suitable message or reminder upon using the system, you can define these here.

`login_banner`	A banner text message to be displayed to anyone who attempts to log in to the Admin UI or Stingray SSH command line.
`banner_accept`	Whether the user is required to explicitly acknowledge and agree to the `login_banner` text prior to logging into the Admin UI. A check box is added to the login page for this purpose.
`uipage_banner`	This is a text message that will be displayed at the top and bottom of *each page* of the Admin UI.

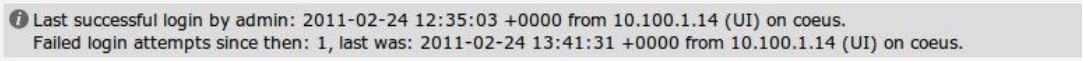
Login controls

You can place controls on the number of login attempts available to users, and the consequences of breaching this.

`max_login_attempts`	The number of login failures permitted before a user account is suspended. It will be reactivated after the delay set in `max_login_suspension_time`, however can be reactivated sooner from the **System > Users > Local Users > Edit** page or the **System > Users > Suspended Users** page (see the *Suspended Users* section of CHAPTER 24).
`max_login_external`	Specifies whether externally authenticated (LDAP, RADIUS or TACACS+) users should be suspended after `max_login_attempts` login failures.
`max_login_suspensio`	The length of time for which a user account is

n_time	suspended after max_login_attempts login failures.
login_delay	The delay after a failed login before another login attempt can be made.

The login information banner



At the top of the Home Page, an information banner describes the previous successful login attempt by the current user on this system. Any previous failed attempts are also shown here, in order that you can identify any unexpected or unauthorized use of your login credentials.

The Audit Log

You can use the Audit Log to monitor recent configuration changes. The audit log records login attempts, configuration changes (by user and source IP address), and user logouts. It also records changes made using the Stingray Control API.

You can find the Audit Log in the **Diagnose** section of the Stingray Admin Server.

Audit logs are automatically rotated once the maximum file size of 50MB is reached for the current log. As the log is rotated, a new archive file is created for the log entries up to that point, and the current log cleared. Archive files are then compressed with 'gzip' and given a filename in the format audit-<date>-<sequence>.gz (where <sequence> is an incremental number provided for multiple same-day rotations).

Archived log files can be found on the **Audit Archive** page (accessed via the *Manage archived audit logs* link). Each can be downloaded or deleted from here using the controls provided.

Note: The maximum file size value can be overridden by setting the environment variable AUDITLOG_ROTATE_THRESHOLD, although certain virtual appliance/cloud variants may have an alternate automatic threshold that is implemented in place of this. This will be detailed in the relevant *Getting Started Guide*.

CHAPTER 25 Stingray Control API

This chapter gives a brief overview of the Stingray traffic management Control API. For further information, please refer to the **Control API Manual** included with your distribution.

Introducing the Stingray Control API

The Stingray Control API is a standards-conformant SOAP-based API that can be used to remotely administer and configure a traffic manager cluster. The Control API provides an alternative to the web-based Admin Server and is suitable if you wish to configure the traffic manager from another application.

For example, when an Intrusion Detection System detected a remote attack attempt, it could use the Stingray Control API to configure the traffic manager cluster to drop all connections from the suspect IP address.

A provisioning system could detect server overloading by monitoring the response times of the server nodes using the traffic manager's Service Level Monitoring capability and the SNMP interface. Once it had provisioned additional servers, it could then inform the traffic manager by reconfiguring the server pools using the Stingray Control API.

The Stingray Control API can be used by any programming language and application environment that supports SOAP services. Perl, Python, Java and C# are commonly used.

Note: The Stingray Control API is not available on all Stingray traffic management solutions. Please refer to your product specifications for details.

Example: listing running Virtual Servers

Perl with SOAP::Lite

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

use SOAP::Lite 0.60;

# This is the url of the Stingray admin server
my $admin_server =
'https://username:password@host:9090';

my $conn = SOAP::Lite
    ->
uri('http://soap.zeus.com/zxtm/1.0/VirtualServer/')
    -> proxy("$admin_server/soap");

# Get a list of virtual servers
my $res = $conn->getVirtualServerNames();
my @names = @{$res->result};

# Establish which are enabled
$res = $conn->getEnabled( \@names );
my @enabled = @{$res->result};

# Print those which are enabled
for( my $i = 0; $i <= $#names; $i++ ) {
   if( $enabled[$i] ) {
      print "$names[$i]\n";
   }
}
```

Run the example as follows:

```
$ ./listVS.pl
Main website
Mail servers
Test site
```

C Sharp or Mono

```
using System;
using System.Net;
using System.IO;
using
System.Security.Cryptography.X509Certificates;

public class AllowSelfSignedCerts :
IcertificatePolicy {
   public bool CheckValidationResult(
         ServicePoint sp, X509Certificate cert,
         WebRequest request, int problem )
   {
      return true;
   }
}

public class listVS {

   public static void Main( string [] args )
   {

System.Net.ServicePointManager.CertificatePolicy =
         new AllowSelfSignedCerts();

      string url= "https://host:9090/soap";
      string username = "username";
      string password = "password";

      try {
         ZXTM.VirtualServer p = new
ZXTM.VirtualServer();
         p.Url = url;
         p.Credentials = new NetworkCredential(
username, password );

         string[] names =
p.getVirtualServerNames();
         bool[] enabled = p.getEnabled( names );

         for ( int i = 0; i < names.Length; i++  )
{
            if( enabled[i] ) {
               Console.WriteLine( "{0}", names[i]
);
```

```
            }
         }
      } catch ( Exception e ) {
         Console.WriteLine( "{0}", e );
      }
   }
}
```

This code works with the .NET 1.1 SDK and with Mono.

Using .Net 1.1, compile and run this example as follows:

```
C:\> wsdl -o:VirtualServer.cs -n:ZXTM
VirtualServer.wsdl

C:\> csc /out:listVS.exe VirtualServer.cs listVS.cs

C:\> listVS.exe

Main website

Mail servers

Test site
```

With Mono, compile and run as follows:

```
$ wsdl -o:VirtualServer.cs -n:ZXTM
VirtualServer.wsdl

$ msc /out:listVS.exe /r:System.Web.Services \

        VirtualServer.cs listVS.cs

$ listVS.exe

Main website

Mail servers

Test site
```

Further examples

For further examples using other programming languages, please refer to the **Stingray Control API Manual**.

CHAPTER 26 Command Line Interface

Your traffic manager is normally managed using the web-based Admin Server. For scripting the configuration, or integrating with other systems, a SOAP-based Control API (CHAPTER 25) is also available.

However, writing an application to use the Control API can take time and may not be appropriate for small, simple tasks or one-off configuration. As an alternative, you can use the command line interface (CLI) for quick access to the full functionality of the traffic manager.

The CLI functions as an interactive shell, and can also be scripted. It uses the Control API to communicate with the traffic manager cluster, and the commands available in the CLI correspond to the methods available in the Control API.

Accessing the CLI

The CLI is started by running the program `zcli` – which can be found in the following location:

- `ZEUSHOME/zxtm/bin/zcli`

You can connect to the CLI by logging in to a virtual appliance/cloud instance directly using SSH (see the *Remote Login to Stingray* section of CHAPTER 22) and typing the command `zcli`. Alternatively, you can connect remotely by specifying the user and host as arguments to the `zcli` command:

- `zcli [user@]host:port  e.g. zcli admin@ztm1:9090`

Usage and options for the `zcli` program can be listed by specifying "`--help`" as an argument:

```
$ zcli --help
Usage: ./zcli [OPTIONS] [user@][host:port] [script
file]
Configure Stingray Traffic Manager from a script
file or standard input.

  --user USER        Set the username (default:
admin)
  --passfile FILE    Read the password from this
file
  --nossl            Your traffic manager does not
have SSL enabled admin server
  --verbose          Verbose output (for testing)
```

```
  --continue         When running a script, continue
even if an error occurs
  --help             Show this help
  --version          Show the version of this
program
  --formatoutput     Use 'human readable' output,
even in script mode
  --json             Use strict JSON output, even in
interactive mode
  --timeout SECONDS  Set timeout for commands

If no host:port is specified, the local traffic
manager will be used.
```

Permissions

The CLI must authenticate itself with the traffic manager using a username and password with 'Control API' permissions (see the *Network Design* section of CHAPTER 23). Any user in the 'admin' group has appropriate permissions, and you can add those permissions to other user groups.

By default, the CLI uses the username 'admin' to communicate with the traffic manager (the `--user` option can be used to select another username):

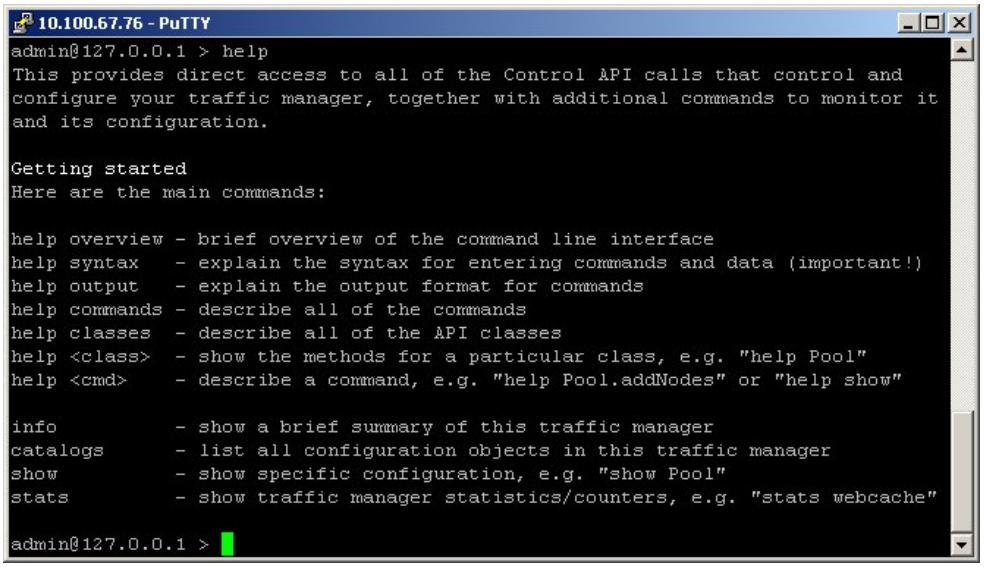
```
$ zcli
Please enter your password for user 'admin' on the
local traffic manager
Connected to 127.0.0.1:9090
admin@127.0.0.1 >
```

Once connected, the CLI displays a prompt (showing the username and hostname of the traffic manager) and waits for a command to be entered:

```
admin@127.0.0.1 >
```

Commands can be typed in one at a time. A command history is maintained, so pressing the up and down arrow keys will step back through previous commands. Auto-completion is also available for many commands; pressing the tab key will auto-complete any half-typed command or object name. Pressing 'tab' twice will list the valid options.

The command help provides a simple overview of the CLI and suggests help topics to find more information on the available commands:



Commands

To discover the available commands, type `commands` for a detailed list, or use tab-completion (press the tab key twice) for a brief list of commands and objects.

Commands fall into three main groups:

1. **Control API methods**

   These are of the form `Class.method [arguments...]`, for example:

```
admin@127.0.0.1 > Pool.getNodes "Web Servers"
[ "92.52.65.222:80", "92.52.65.213:80" ]
```

2. **Helper commands**

   These summarize the traffic manager configuration; the main commands are:

   - `info` - show a brief summary of this traffic manager
   - `catalogs` - list all configuration objects in this traffic manager
   - `show` - show specific configuration, e.g. `show Pool`

- `stats` - show traffic manager statistics/counters, e.g. `stats webcache`

3. **CLI help**

   - `help syntax` - explain the syntax for entering commands and data
   - `help output` - explain the output format for commands
   - `help commands` - describe all of the commands
   - `help classes` - describe all of the API classes
   - `help <class>` - show the methods for a particular class, e.g. `help Pool`
   - `help <cmd>` - describe a command, for example, `help Pool.addNodes` or `help show`

Control API methods

All of the Control API methods are accessible via the CLI. The corresponding command is constructed from the class name and method name as listed in the Stingray Control API manual; refer to this manual for details of each command.

For example, to list all of the virtual servers:

```
admin@127.0.0.1 >
VirtualServer.getVirtualServerNames
[ Intranet, "Secure Site", webmail ]
```

Understanding arguments to CLI commands

Arguments for commands are given as space separated names; commas between arguments are optional. Argument values that contain a space or other special characters that may confuse the syntax should be "quoted". Lists are contained within '[' and ']' square brackets.

For example, the Control API document for the traffic manager details the method 'getNodes' in the Pool class as follows:

getNodes(names) throws ObjectDoesNotExist

Get the lists of nodes for each of the named pools.

```
String[][] getNodes(
```

```
                String[] names
            )
```

The corresponding CLI command would be `Pool.getNodes`. The documentation shows that the command takes an array of pool names (`String[] names`); the command line to call this method on two pools would be:

```
admin@127.0.0.1 > Pool.getNodes [ pool1 pool2 ]
{ pool1: [ "serverA:80", "serverB:80" ],
  pool2: [ "serverC:80", "serverD:80" ] }
```

The output from `Pool.getNodes` is declared as `String[][]`, i.e. an array of arrays. For each pool that was specified, an array of nodes is returned.

For convenience, in its interactive mode, the CLI indexes the output of the command to show the list of nodes prefixed by the pool name that they refer to.

Using a single argument

Many commands in the Control API take a list of arguments so that they can operate on more than one object at once. To simplify the use of commands with a single item, the array can be omitted:

```
admin@127.0.0.1 > Pool.getNodes pool1
[ "serverA:80", "serverB:80" ]
```

Using wildcards

The '*' symbol is a wildcard that can be used to match multiple objects:

```
admin@127.0.0.1 > VirtualServer.getDefaultPool *
{ Intranet:      pool1,
  "Secure Site": "Secure pool",
  webmail:       "Mail servers" }
```

Partial names can also be wildcarded. For example, to enable keep-alives on all the pools with a name beginning 'Pool':

```
admin@127.0.0.1 > Pool.setKeepalive Pool* 1
```

Complex Argument types

Some commands require more complicated argument structures. For instance, `VirtualServer.addVirtualServer` is defined in the Control API manual as:

```
void addVirtualServer(
   String[] names,
   VirtualServer.BasicInfo[] info )
```

The `BasicInfo` structure is defined as:

```
struct VirtualServer.BasicInfo {
   # The port to listen for incoming connections
on.
   Integer port;

   # The protocol that this virtual server handles.
   VirtualServer.Protocol protocol;

   # The default pool that traffic to this virtual
server
   # will go to.
   String default_pool;
}
```

Structures are supplied to the CLI by using '{' and '}' curly brackets, identifying each member with key:value pairs. For example,

```
admin@127.0.0.1 > VirtualServer.addVirtualServer
Intranet { port: 80, protocol: http, default_pool:
pool1 }
```

If you need help with the expected arguments for any command, you can use the help available through the CLI:

```
admin@127.0.0.1 > help
VirtualServer.addVirtualServer
```

Built-in commands

The CLI includes a number of additional commands that simplify viewing and modifying configuration. Full help and usage instructions can be found using `help <commandname>`.

info

Shows high-level information about this traffic manager, similar to the Home Page of the Admin UI. This shows the traffic managers in a cluster, the virtual servers and pools configured, traffic IPs, any configuration or operational errors, and the most recent entries in the event logs.

connect

Connect to a different traffic manager. The CLI can be used to communicate with remote traffic managers, including traffic managers that are not in the same cluster as the local machine. This command is also useful when writing scripts, as the traffic manager hostname, username and password can all be contained in the script file.

errorlog

Shows the most recent messages in the traffic manager event logs.

catalogs

Lists the contents of the configuration catalogs, showing items such as Monitors, TrafficScript Rules and Java Extensions.

show

Show the configuration of any configuration object in the traffic manager. For example:

- `show VirtualServer Intranet` - shows a summary of the virtual server 'Intranet'
- `show Pool` - shows a summary of all pools
- `show VirtualServer customer*` - shows a summary of the virtual servers customer*

You can also just use show with an object name, and it will summarize all objects that match that, for example:

- `show Staging` - shows a summary of any object called 'Staging'
- `show *` - shows everything (may take some time)

The full list of possible object types is:

- `action, authenticator, bandwidth, ca, certificate, clientcertificate, connections, event, global, info, java, licensekey, monitor, persistence, pool, protection, rate, rule, slm, trafficip, virtualserver`

stats

Shows counters and statistics for various different objects in the traffic manager. For example:

- `stats VirtualServer Web` - shows stats for the virtual server 'Web'
- `stats Pool` - shows stats for all pools
- `stats *` - shows stats for everything

You can also just use stats with an object name, and it will display stats for all objects that match that name, for example:

- `stats Staging` - shows stats for any object called 'Staging'
- `stats *` - shows stats for everything (may take some time)

The full list possible stats objects is:

- `authenticator, bandwidth, event, interface, node, perpoolnode, pool, protection, rate, rule, session, slm, slmpernode, ssl, system, trafficip, virtualserver, webcache, zxtm`

help

- `help overview` - brief overview of the command line interface
- `help syntax` - explain the syntax for entering commands and data
- `help output` - explain the output format for commands
- `help commands` - describe all of the commands

- `help classes` - describe all of the API classes
- `help <class>` - show the methods for a particular class, e.g. `help Pool`
- `help <cmd>` - describe a command, for example, `help Pool.addNodes` or `help show`

rule

Upload, download and syntax check TrafficScript rules on the traffic manager.

getbackup

Download a backup from the traffic manager.

putbackup

Upload a backup file to the traffic manager.

techsupport

Download a technical support report, for use when contacting support.

download

Download a specific configuration file from the traffic manager, for offline editing or archiving.

upload

Upload a specific configuration file to the traffic manager.

edit

Download and edit a specific configuration file from the traffic manager, and then upload the modified version.

watch

Repeatedly run a command and watch the output. This is useful for monitoring changes, e.g. keeping an eye on logs or recording stats. For example:

```
admin@127.0.0.1 > watch 10 stats webcache
```

This will show all of the webcache statistics, updated every ten seconds.

Scripting the CLI

The CLI normally runs in 'interactive' mode, where commands can be entered from the keyboard and are run instantly.

It is also possible to run a script of CLI commands without user input. For example, the following script prints out the webcache statistics, clears some items from the cache and then prints the webcache statistics again:

```
System.Stats.getWebCacheEntries
System.Cache.clearCacheContentItems "Intranet"
"http" "www.stingray.com" "*/product/*"
System.Stats.getWebCacheEntries
```

Save the script to a file on the local machine, then execute it as follows:

```
$ ZEUSHOME/zxtm/bin/zcli /home/user/webcache-script
```

The CLI will read the file and run each command found in turn. If an error occurs when running a a command, the script execution will stop and the CLI will exit.

To fully automate a script, the username and password must be specified. This can be achieved in two ways:

1. Store the password in a file and provide this file to the `zcli` script:

```
$ ZEUSHOME/zxtm/bin/zcli --user username --passfile
/home/user/password /home/user/webcache-script
```

   The file containing the password should have suitably restricted permissions so that other users cannot read it.

2. Use the CLI `connect` command

Add the `connect` command to the start of the script, so it becomes:

```
connect localhost 9090 username password
System.Stats.getWebCacheEntries
System.Cache.clearCacheContentItems "Intranet"
"http" "www.stingray.com" "*/product/*"
System.Stats.getWebCacheEntries
```

Script output

The output from scripted CLI commands is slightly different from interactive CLI commands. In interactive mode, the output is formatted and spaced to make it more human-readable, spreading the output over several lines if needed. In scripted mode, the output from each command is written on one line only, without any extra padding or formatting.

The output format is JSON-formatted, to facilitate importing and parsing in other programs. Output from commands that take several inputs (e.g. multiple pool names) is not presented in a table to highlight the results for each item.

For example, compare the output from the same command run on interactive and scripted sessions.

Interactively:

```
admin@127.0.0.1 > Pool.getNodes [ pool1 pool2 ]
{ pool1: [ "serverA:80", "serverB:80" ],
  pool2: [ "serverC:80", "serverD:80" ] }
```

From within a script:

```
$ ZEUSHOME/zxtm/bin/zcli --user username --passfile
/home/user/password
/home/user/getnodes
[["serverA:80","serverB:80"],["serverC:80","serverD
:80"]]
```

If you want to keep the human-readable output, then run `zcli` with the `--formatoutput` argument.

CHAPTER 27 Granular Configuration Import/Export with zconf

Introduction

If you are likely to provision a significant number of traffic managers on a regular basis, any way to minimize duplication of effort during the provisioning process is a way to decrease time-to-deployment costs. Quite often, users will have a standard set of policies or common pieces of configuration that they want to roll out to a large number of services. From time to time, one may wish to make the same configuration change across a large enough number of traffic managers.

Users may have some kind of change control database to store policies or configuration files. Subversion (SVN) is an example of a tool that can be used to store and manage changes to not only source code, but configuration files as well. Stingray includes a tool called `zconf` that can be used to perform granular levels of configuration export and import and to store those [partial] configurations in a tarball. If desired, the tarball can be extracted and checked into one of these change control databases.

Using zconf

The `zconf` utility can be found in `$ZEUSHOME/zxtm/bin`, where `$ZEUSHOME` is the directory that you installed the traffic manager to. Note that if `$ZEUSHOME` is not set correctly that the `zconf` utility will not function correctly. This command line utility has a built-in help system. When you run the program with no arguments, you should see the following output (this example taken from the Stingray virtual appliance):

```
root@ubuntu1:~# $ZEUSHOME/zxtm/bin/zconf

ERROR: No command specified.

Usage: /opt/zeus/zxtm/bin/zconf [COMMAND] [OPTIONS]
[ARGUMENTS]
Commands:

  copy
  diff
  export
  getdefaultfilter
  help
```

```
  import
  info
  list
  replicate
  version

For more information on a given comment execute:

 /opt/zeus/zxtm/bin/zconf help <COMMAND>
```

This will show you a list of commands. To learn how to utility works, you should read the help for each command.

Exporting a Complete Backup

If you want to take a complete backup of your traffic manager's cluster configuration, use `zconf`'s export command. By default, the output of the export command is printed to STDOUT. If you want to save the output to a file, you can use either the `--file` argument, or redirect the output from STDOUT to a file (as in the example below). On most Linux or Solaris systems, you can then use the `tar` command to extract the contents of the tarball (i.e. for checking in to a subversion repository, for example).

```
root@ubuntu1:~# $ZEUSHOME/zxtm/bin/zconf export >
backup.tar
root@ubuntu1:~# ls
backup.tar
root@ubuntu1:~# tar xf backup.tar
root@ubuntu1:~# ls
backup.tar  conf
root@ubuntu1:~# ls conf
actions          groups         monitors  security
users
activitymonitor  licensekeys    PARTIAL   services
VERSION_8.0
commkey          locations      rules
settings.cfg  zxtms
events           locations.cfg  scripts   TIMESTAMP
root@ubuntu1:~#
```

Configuration Listings

Configuration backups are stored in a number of plain-text files and organized into a directory tree. The `list` command of `zconf` can be used to list the configuration files in the current running traffic manager and any configuration backup or backup archive. Say for example that you want to get a list of the virtual servers configured on the current traffic manager:

```
root@ubuntu1:~# $ZEUSHOME/zxtm/bin/zconf list
vservers
root@ubuntu1:~#
```

This output indicates that the `vservers` directory is empty, and consequently that there are no configured virtual servers. Suppose you want to list the virtual servers in a configuration archive:

```
root@ubuntu1:~# $ZEUSHOME/zxtm/bin/zconf list
vservers          --conf-path backup.tar
monitors/Full HTTP
pools/Bar
vservers/Foo
```

Notice that while you specified `vservers` as the glob argument to the `list` command, that a pool and a monitor are also listed in the output. This is because all dependencies required to make that virtual server work as expected are printed.

Partial Imports

Assuming that you have a configuration archive tarball in the current directory, in order to import a virtual server (called "Foo" for example) and all of its dependencies you should use the following command:

```
zconf import --include vservers/Foo backup.tar
```

This import will cause the configuration to be replicated across the other members of the cluster.

Note: Similar functionality is also available from within the Admin UI, on the **System > Backups > Partial Backups** page. Please refer to the *System > Backups > Partial* section of CHAPTER 7 for more details.

CHAPTER 28 Multi-site cluster management

Introduction

The multi-site cluster management functionality discussed in this chapter builds on the sophisticated capabilities of Stingray by adding support for management of *multiple* distributed physical, virtual or cloud-based traffic manager clusters. These clusters could be present at various geographic locations, on various platforms and include varying numbers of traffic manager instances. Each might host specific services that you want to administer from one central location.

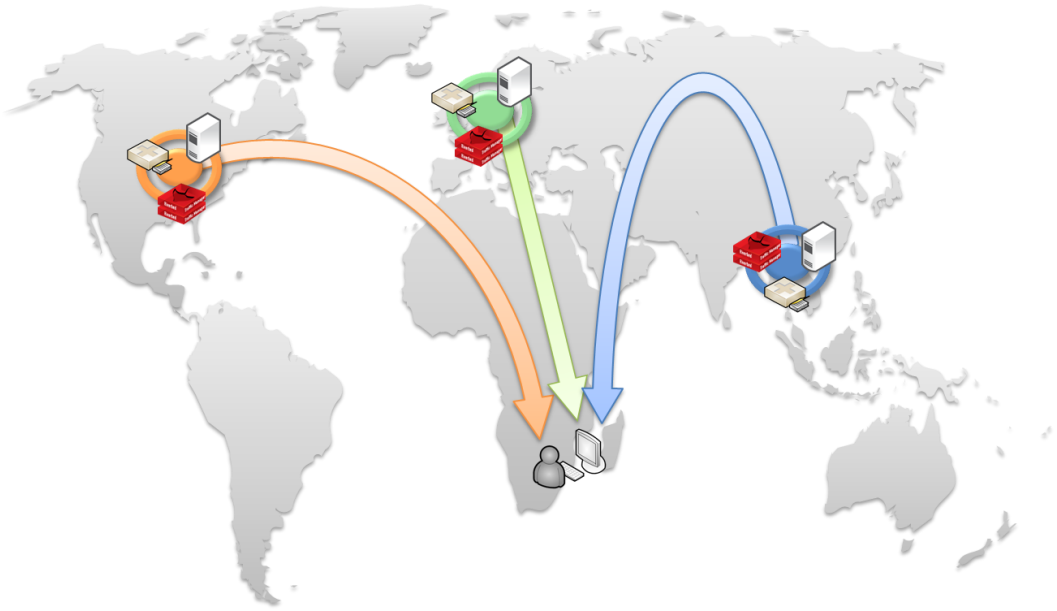


Fig. 41. Centrally managing your global traffic manager cluster infrastructure

This chapter discusses how your globally-distributed Stingray infrastructure can be pulled together into one centrally-managed cluster - whilst retaining the location-specific configuration of your traffic manager systems.

This feature is not to be confused with Global Server Load Balancing (GSLB), which involves geographically-aware load balancing across services hosted at multiple locations around the world. The implementation of GSLB in Stingray is discussed in CHAPTER 29 below.

This chapter assumes you are familiar with the Stingray user interface and general configuration concepts described elsewhere in this manual. It is strongly recommended that you refer to CHAPTER 7 before continuing.

Activation and deactivation

You can enable multi-site management functionality from the **System > Traffic Managers** page of the Admin UI. Simply tick the **Confirm** checkbox and then click **Enable** in the *Multi-Site Management* section to activate Stingray in this mode.

Equally, Stingray can be set back to its original state by deactivating the multi-site capability in the same way. In this scenario, you must first select one of your defined configuration locations as the *master* configuration to be adopted by all traffic managers. All other configuration locations are then removed along with any location-specific configuration values stored for them. Ensure your chosen location is selected in the drop-down list, tick the **Confirm** checkbox and then click **Disable**.

Note: Whilst configuration locations are removed, GLB locations remain - but in a deactivated state.

Key concepts

Configuration Locations

Configuration locations are essentially groupings of traffic managers, perhaps situated in a particular physical location, that are required to host the same services within an overall cluster. Configuration locations are defined within the **Catalogs > Locations** page, and each of your traffic managers are then marked as present at a particular one.

Throughout the Admin UI, services and other configuration aspects can be made location-specific, and only those traffic managers marked as present at that location will use them.

This concept provides the ability to combine all of your globally-located Stingray application delivery requirements into one centrally administered system. From a single Admin UI, you can now configure and monitor all your services regardless of the complexity of the setup.

Note: It is not mandatory to have your traffic managers physically present at a particular location. The concept of a configuration location, as opposed to a GLB location, is mainly as a mechanism to allow virtual grouping within your traffic manager cluster.

Clusters

Within a multi-site configuration, a cluster is a group of Stingray traffic managers deployed over one or more global locations, centrally managed, in order to provide the delivery of application services in a fault-tolerant manner.

Stingray service configuration is replicated between all machines in the cluster. As with non-multi-site environments, the majority of the configuration you make is shared, however multi-site management extends this facility by providing the ability to set location-specific configuration that is only active on those traffic managers marked at that location. This provides a form of sub-clustering, or local service delivery. You can edit the overall configuration using the Admin UI of any Stingray Traffic Manager in your cluster and it will be replicated to all other traffic managers.

A small amount of system configuration is specific to each location (such as local DNS servers), and a small amount of system configuration is specific to a particular machine (such as SNMP settings). To configure location-specific system information, you can use the UI of one of the Stingray Traffic Managers in that location, and it is replicated to other Stingray Traffic Managers defined at the same location. To configure machine-specific system information, you must use the UI on that machine.

Important: Where you are intending to join a remotely-located traffic manager into a cluster, it is essential that you provide it an **External IP address** first. Furthermore, it is recommended that you ensure the availability of this address, and in particular ports 9080 and 9090, across your cluster prior to adding the new machine. Failure to observe this could cause configuration replication problems within your cluster. See the *Setting traffic manager locations* section of this chapter for more details on how to set the External IP.

Deployment scenarios

Note: All references to 'location' in this section refer to **Configuration Locations**. Additionally, all scenarios assume suitable IP/port connectivity to external locations (see above).

Create & manage a second Stingray location

As your services grow in popularity and sophistication, you may wish to deploy a second Stingray presence in some other global location to be run concurrently.

- First, perform installation and any initial configuration required by each of the new traffic managers at the new location. Please note

however that when joining an existing Stingray cluster, any non-system specific configuration will be overwritten as the cluster config is replicated to each new machine being added.

- From the Admin UI of each traffic manager in turn, run the **Join a Cluster Wizard** to connect to your existing Stingray cluster.

- Either pre-define the new second configuration location on your existing cluster, or specify the 'new location' option on step 4 of the wizard. Note that specifying a new location during the wizard will not apply a value to the *position* setting in the location definition. You will need to do this manually.

- For each subsequent traffic manager, whilst running the wizard you should ensure you select the newly created location on step 4.

Each new traffic manager will now be added into the cluster, with a copy of the full configuration, but marked as present at the second configuration location. Any configuration changes that were, or are subsequently, made in a location-specific way will be applied across the whole cluster.

Add a new traffic manager to your multi-site cluster

You can add additional traffic managers to any location managed by Stingray by running the **Join a Cluster Wizard** from the Admin UI of the machine you are adding. This will import the configuration of your cluster to your new traffic manager and it will appear as a managed machine within the Admin UI.

Merging two or more existing Stingray clusters

In normal cluster joining operations, the joining machine takes on the configuration of your cluster, thus deleting any existing configuration you might have had. What we are attempting to do here is to merge two or more distinct configurations so special care must be taken to preserve the settings on all machines.

Stingray Traffic Manager provides a command line tool 'zconf' to perform fine-grained config import/export operations. With this tool we can choose to import the individual services of the secondary clusters into the master cluster config without compromising the existing services. It is also worth making use of *zconf*'s built-in "dry-run" mode for any import procedures, to ensure that you are fully aware of the expected outcome. For full usage syntax and further information, please refer to the online help available directly from *zconf*.

The intention is to minimize service downtime as best as possible, however it is important to note that there may be service interruptions as Traffic IP addresses are transferred so this is best performed at a time where your web traffic is at its least.

In all cases, it is strongly recommended you backup the configuration on all clusters before attempting this procedure. It is also worth attempting the following with a non-critical system first, in order to satisfy yourself of the likely outcome.

- Ensure all your traffic managers in all clusters are running the latest Stingray software. Designate one cluster as the *master* cluster.

- Set up any new secondary configuration locations on your master cluster.

- In order to minimize filename clashes, you may wish to rename your virtual servers, rules, pools, etc., on all clusters to uniquely identify them, such as by pre-pending the location name. For example, renaming "Intranet" to "london-Intranet" and so on. This is not mandatory, however it should reduce the chances of config clashes. The default behavior of *zconf* is to overwrite existing same-name config files when you perform an import, however this is switchable.

- Export your remote cluster configuration using the **System > Backups** page, and copy the resultant tarball over to your master cluster.

- Using *zconf*, import the virtual servers (`/vservers`) directory from your config backup. *zconf* will automatically follow dependencies in each virtual server file and import in the relevant supporting files. The only caveat with this is the use of any Java extentions - you may need to manually import in the `/jars` and `/servlets` directories yourself.

- These imported services should now be marked as enabled only for the newly defined remote location. This can be achieved on the **Virtual Server > Edit** page by clicking the *Edit setting by location* icon next to the *Enabled:* setting. Ensure this virtual server is enabled for the remote location and disabled for all other locations.

- Use of Traffic IP Groups can cause service conflicts if you have the same IPs set up on two clusters. You may need to take the IPs in your existing remote cluster offline in order to set up the same IPs in your master cluster. As you add in your remote traffic managers to the master cluster, you will need to first remove them from the remote cluster, and as a result, any remote IP groups they appear in first.

- Assuming you have multiple traffic manager machines in each cluster, you can disconnect a single traffic manager from each and join it to the master cluster, marked as being at the corresponding defined remote location. **Please note that system specific settings should be applied directly through the Admin UI of the machine they pertain to.**

- This traffic manager can be added to the newly created Traffic IP Group and your services should resume.

- Now you can take each remaining remote traffic manager out of the remote cluster and join them into the master cluster, marked as being at the appropriate location and added to the correct Traffic IP Group. Each will inherit the fully merged configuration, and yet will only respond to the services marked for that location.

Configuration

Setting up locations

Locations are listed and modified from the **Catalog > Locations** page. You can also find them through the **System > Traffic Managers** page, by clicking on the *Manage Locations* link. Your defined locations are listed here, with the opportunity to add to, and remove from, the list.

Note: For product variants that include **Global Load Balancing**, a separate additional section is shown for "GLB Locations". Please see CHAPTER 29 below for more details.

Each location is simply defined by its name. Clicking on the location name or associated **Edit** link allows you to modify its settings or delete the location altogether.

Adding a new location can be achieved using the *Add new Location* section under the main list. All new locations are created using an existing location as a template (excluding the case where there are no existing locations). **This 'Based on' mechanism copies any location-specific config for the template location to your new location.**

Locations can be deleted from the section at the bottom of the associated Edit page. Note however that locations in use by one or more traffic managers cannot be renamed or deleted. You must first disassociate your traffic managers with this location before such operations can be performed.

Setting traffic manager locations

On the **System > Traffic Managers** page, each traffic manager listed in your cluster has two new configurable settings:

- **Location:** Select this traffic manager's configuration location from the drop-down list.

- **External IP:** Where your traffic manager's host name is not globally resolvable, you will need to specify an externally available IP address for communications between your cluster members.

Any changes to these settings must first be confirmed by clicking the checkbox next to the **Update** button.

Location-specific configuration

A significant change to the way Stingray handles configuration settings whilst in multi-site mode can be seen by the addition of a new icon next to each individual setting.

Fig. 42. The Location icon

Clicking this icon separates out the setting into multiple instances, one for each configuration location. Clicking the icon again will revert the setting back to being a singular instance - which is used by all locations.

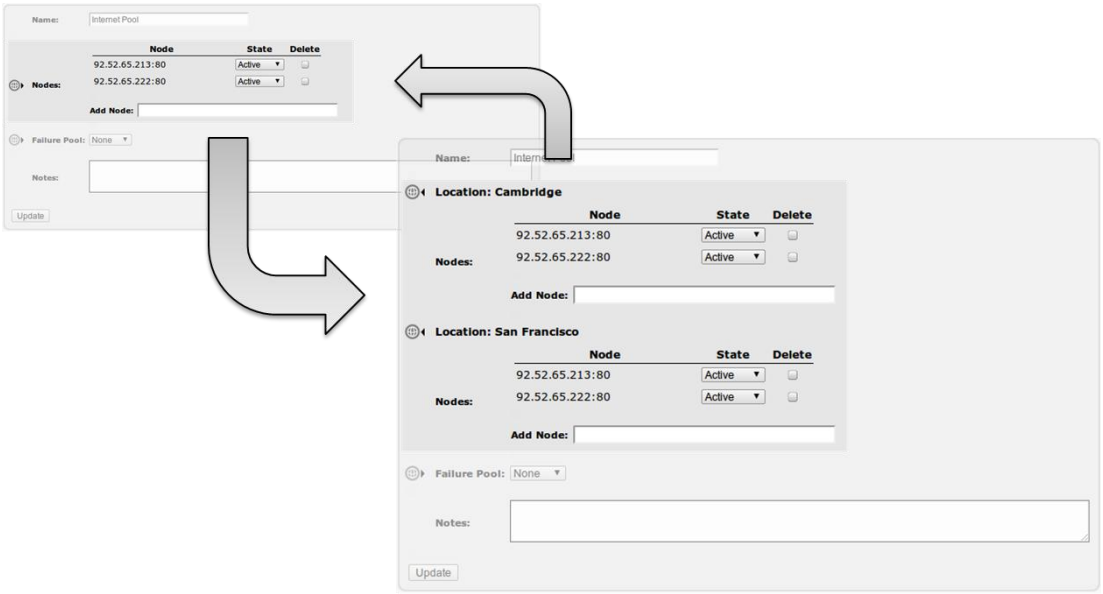


Fig. 43. Switching your node list between location and normal mode

This mechanism allows you to specify individual configuration settings on a per-location basis, whilst maintaining some global settings that apply to all cluster members.

Home page changes

When multi-site cluster functionality is enabled, a modified admin UI home page is presented. The new style is more suited to the environment within which Stingray is expected to operate. It now shows a number of distinct sections as listed below:

- **Locations:** Your traffic manager configuration locations are shown here. Locations defined but not in use will not be shown. Click on the location name to view the **Catalogs > Locations > Edit** page for that location.

- **Services:** This shows the list of services marked as *enabled* and managed by this Stingray installation. A service can be considered as a combination of virtual server, rules, pools and other associated objects. Click on the service name to view the **Virtual Servers > Edit** page for that service.

- **Event Log:** This is a quick summary of the event log. It is written-to by all traffic managers with the name of the machine in question alongside each entry. The full logs can be viewed by clicking the *Examine Logs* link.

Each of the locations and services are displayed with a suitable status indicator:

Indicates running ok

Indicates a warning

Indicates a serious error

Warnings are non-critical problems that mean the location or service is still running, but with impared performance. Errors require action to rectify a problem that is preventing your locations or services from operating. The status applet will reflect these conditions. In the case of warnings or errors, clicking on the location/service name takes you to the **Diagnose** page, where you will see a full explanation of the problem.

The world map

Fig. 44. The world map

Click on **Activity > Map** to show a world map for enhanced visualization of your locations and services. You can also view the map via the *View traffic on World Map* link from the **Virtual Servers > Edit** page. This provides a global overview of the locations defined on your system.

The map is designed to be interactive and tools are provided to move around and zoom into a level of detail required for your infrastructure. You can centre the map on a geographic location of your choice by holding down the left-hand mouse button and dragging the map to your required destination. You can show traffic for a specific virtual server using the drop-down control at the bottom. The default is for *all* virtual servers.

Each colored target represents a location, and each small dot that appears on the map represents a user who has issued a request for a particular service. Each user is colored according to the location at which the requested service is hosted. In order to show meaningful information, you will need to enable Connection Archiving (also required for viewing advanced connection analytics). If this is currently disabled, you can click on the *connection archiving is disabled* link at the bottom to enable it. This link takes you to the **System > Global Settings** page where you will need to provide a suitable value, e.g. 500, for the **recent_conns** setting (the default of 0 disables connection archiving).

Traffic Visualization

As with Global Load Balancing, activating multi-cluster management functionality on your system automatically enables additional options into Stingray's existing diagnostic and visualization tools. Please refer to the *Traffic Visualization* section of CHAPTER 29 for more details.

CHAPTER 29 Global Load Balancing

Introduction

Stingray includes a capability entitled Global Load Balancing, designed to provide business continuity and improved performance across globally-distributed locations. This capability is based upon a technique known as GSLB (Global Server Load Balancing).

Geographic load balancing and failover functionality is enabled to allow the distribution of traffic across these locations based on availability, performance and user-defined rules. It is easy to deploy and gives rich feedback on site performance and traffic distribution.

Note: This functionality may not be available in your Stingray product variant. Please contact your service provider who can assist you further.

This chapter describes how to configure Global Load Balancing on your traffic manager. It assumes that you are familiar with standard system administration tasks, and that you are familiar with networking and DNS administration. For a background on DNS, please read the DNS primer below.

Further details on the principles of GSLB are provided below, however please also refer to the background whitepapers and documentation published on www.riverbed.com and the Stingray Community website at http://community.riverbed.com.

DNS Primer

Introduction

The **D**omain **N**ame **S**ystem is essentially the phonebook of the internet. DNS is used to look up different bits of information about domain names. For instance, your web browser will need to know which IP address to connect to in order to request a website from a web server. Similarly, when you send an E-Mail to someone, your E-Mail client will need to know both the domain name and the IP address of the mail server responsible for delivering the message to its recipient. DNS provides the means with which to find this information.

The Tree Analogy

The layout of DNS resembles a tree. At the bottom you have a "root". Higher up you have branches and leaves. If you were to flip the tree upside down, the root

would be on the top. The root of DNS is also at the top of the tree. The root in this case consists of a group of *Root Name Servers*. Below the root are the *Top Level Domain Servers*. The name "com" (as in "www.example.com") or "net" are both examples of *Top Level Domains*. Below the *Top Level* there is no specific classification beyond the term "*Name Server*".

Delegation of Authority

"Authority" in the context of DNS refers to the responsibility of a name server to handle requests for a given domain. If a name server is *Authoritative* for servicing DNS requests for anything under example.com, it has been delegated the authority to do so by the higher-level domain server(s).

Name Resolution

The process of "looking up a piece of information" using DNS is called "*Resolution*". For example, when your web browser needs to find the IP address for a website, it will attempt to *resolve* the domain name to get the IP address. When your E-Mail client needs to know which mail server is responsible for delivering a message to a particular domain, it will attempt to *resolve* the *Mail Exchange (MX)* name, and its IP address.

As part of its network configuration settings, a computer will know of at least one *Name Server*. That *Name Server* in turn provides the computer with *responses* to DNS *requests*. Only under *special circumstances*[12] will a computer send DNS *requests* to a different *Name Server*.

Resource Records

Each piece of information for a domain name is held in what is called a Resource Record. Generally speaking, Resource Records are distinguished by type. For instance, when your web browser wants to know the IP address for www.example.com, it will send a query for an "A record" that matches www.example.com. If your mail client needs to know where it should send a message addressed to john.smith@example.com, it will send a query for an "MX record" that matches example.com. Some commonly used resource record types are:

- **SOA (Start of Authority)**: Provides a few pieces of useful information about a given domain.

[12] As a practical example this would be at the hands of a DNS administrator who is testing his or her work.

- **NS (Name Server):** Indicates which name servers are authoritative for a given domain. This is how authority is delegated from a higher to a lower level domain.
- **MX (Mail eXchange):** Indicates the domain name of the mail server(s) that should accept mail for a particular domain.
- **A (Address):** Contains the IP address of a domain name.
- **CNAME (Canonical Name):** Contains an alias for a domain.
- **PTR (Domain Pointer):** Used for reverse lookups, from IP back to DNS name.

Zone Files

Resource Records are typically stored in text files on authoritative DNS servers. They may also be stored in a database. A zone file contains a set of records which are specific to a domain. Each zone file must contain at least an SOA record, and an NS record. Below is an example of a zone file:

```
$TTL 60 ; 1 minute @

@ IN SOA        ns1.example.com. example.com. (
                                2007080602 ; serial
                                21600      ; refresh (6 hours)
                                3600       ; retry (1 hour)
                                604800     ; expire (1 week)
                                60         ; neg cache ttl (1 minute)
                                )
@         IN                    NS       ns1
@         IN                    NS       ns2

@         IN       MX           10     smtp.example.com.

ns1                             A         212.44.21.73
ns2                             A         65.23.158.26

smtp                            CNAME     www

www                              A         65.23.158.26
```

The Resolution Process

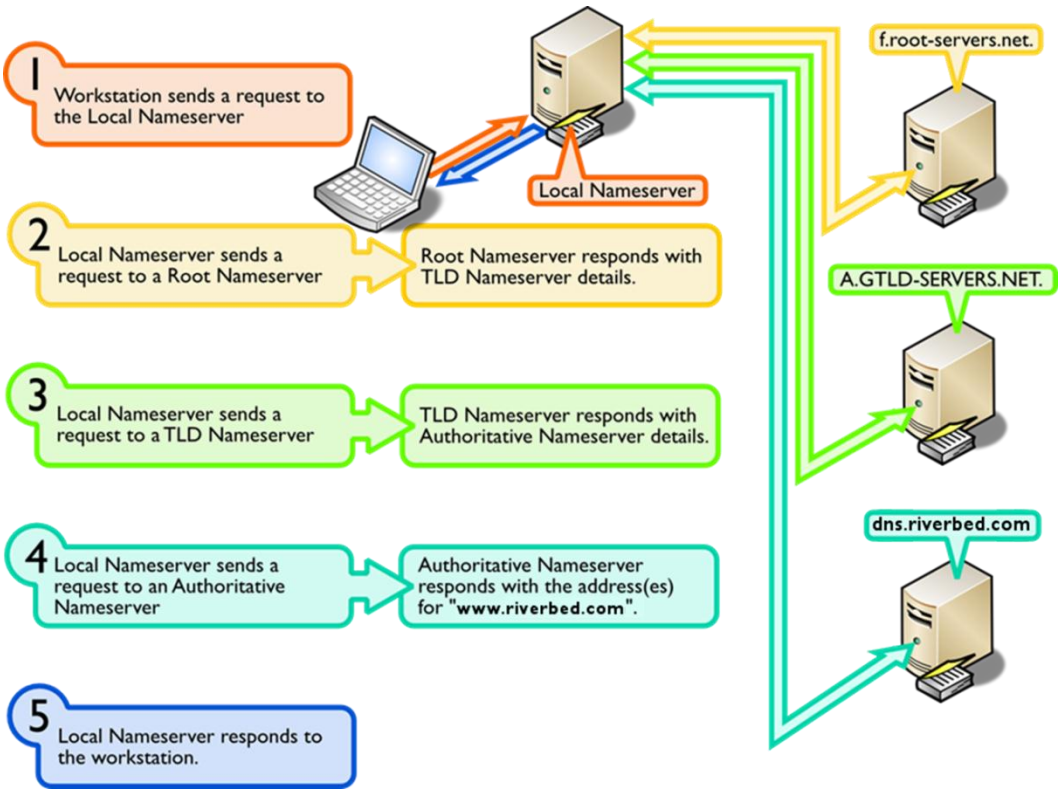


Fig. 45. The Resolution Process

1. When a user opens a web browser and types www.stingray.com into the address bar, his workstation will need to send an HTTP request to a *Web Server*. Before it does so, it must resolve the domain name of the web server. His workstation will send a DNS request to his *Local Name Server* in order to determine the IP address.

2. The *Local Name Server* will then refer to its *Root Hints* zone file. The *Root Hints* zone file contains a list of domain names and IP addresses for the *Root Name Servers*. The *Local Name Server* then sends a DNS request for www.stingray.com to one of the *Root Name Servers* listed in its *Root Hints* zone file. "f.root-servers.net." will compare the DNS request to the list of zones that it is authoritative for. In this case, it is authoritative for the "." domain. Within the zone file for ".", there are delegation records for all *Top Level Domains*. The *Top Level Domain* for www.stingray.com is ".com.", so "f.root-servers.net." sends a DNS response back to the *Local Name Server* containing the domain names

and IP addresses for the *Name Servers* which are authoritative for ".com."

3. A.GTLD-SERVERS.NET[13] is listed (among others) as authoritative for the ".com." domain, so the *Local Name Server* then sends the same request to it. "A.GTLD-SERVERS.NET then responds with a list of name servers authoritative for "stingray.com.".

4. Since "dns.stingray.com." is an Authoritative Name Server for the "stingray.com." domain, the local name server then sends *it* the request. Because "dns.stingray.com" contains all of the *Resource Records* for "stingray.com." in its *Zone Files*, it provides the IP addresses (in the form of an "A" record).

5. Now that the *Local Name Server* has the IP address for www.stingray.com, it passes on the information to the workstation which made the initial DNS request. The machine that performed the lookup is the Local Name Server.

The type of query that the client made is known as "recursive" due to the fact that the local name server searches the DNS tree from top-to-bottom. A "non-recursive" query may also be sent. This means that only a name server authoritative for the domain name being asked about should provide the answer; otherwise, the name server receiving the query should return a list of *Root Name Servers*.

[13] "GTLD" is an acronym for "Global Top Level Domain".

About Global Server Load Balancing

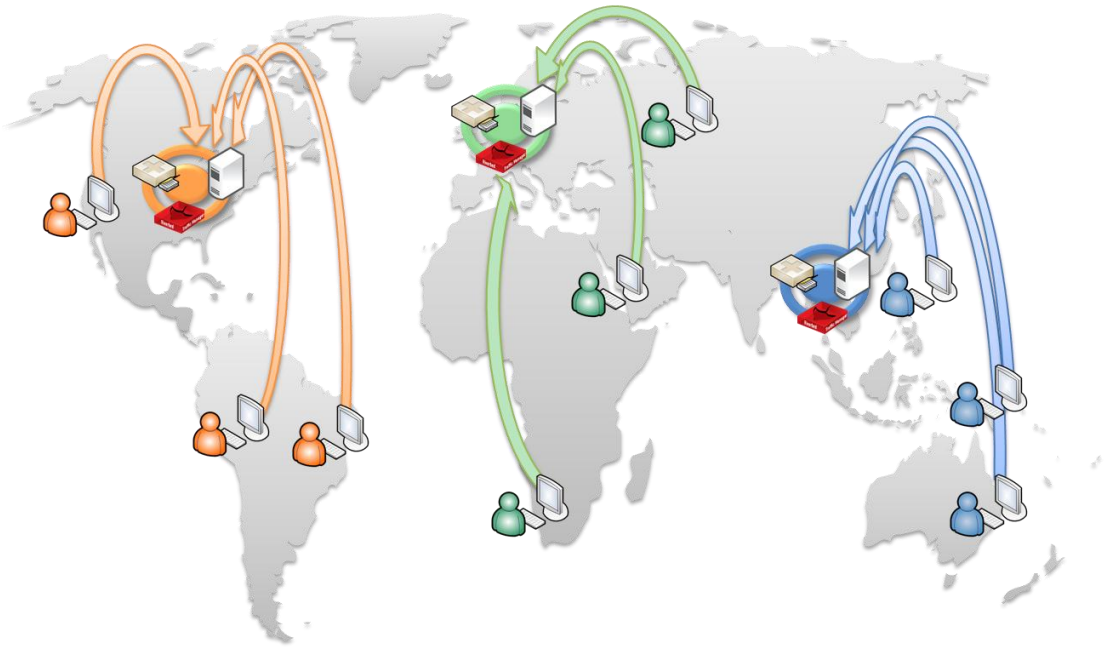


Fig. 46. Global Server Load Balancing

Global Server Load Balancing (GSLB) is a technique that manages how clients are connected to a particular geographical location when a service is hosted in multiple locations.

- In an **Active-Passive** configuration, one location is nominated the active one for each service. The other locations are idle for that service. If the active location becomes unavailable, one of the passive locations becomes active and all clients are directed to it.
- In an **Active-Active** configuration, all locations are used and clients are load-balanced between them based on location performance and proximity.

The primary purpose of a GSLB system is **Business Continuity** - to ensure that services are always available, even when one or more service locations becomes unavailable.

A second purpose of GSLB is to **Improve Customer Experience** - to load-balance each user to the best location from a choice of several. The choice can be based on location performance and proximity, so that clients are directed to the location that is closest and is performing the best. This way, the client gets the best possible level of service.

DNS-based GSLB

The Internet's DNS (Domain Name System) is a little like a large, automated telephone directory, converting domain names (e.g. www.example.com) to IP addresses (like 194.23.1.2). Its operation is largely concealed from end users.

When a service is hosted in multiple locations, each instance of the service will be available from a different IP address. However, an end user will use a single domain name to access the service.

Stingray manipulates the DNS resolution of this domain name to control how each user is directed to a location. It monitors the performance and availability of each location to inform the load-balancing decisions it makes.

Global Server Load Balancing within Stingray

Stingray implements GSLB techniques as the Global Load Balancer (GLB) feature.

In this situation, Stingray works with a standard round-robin DNS configuration. The DNS servers are configured to return the IP addresses of all of the locations hosting a service, and Stingray operates as a proxy in front of these DNS servers. It rewrites the round-robin DNS responses, ensuring that the client is directed to the single most appropriate location.

Before configuring Stingray to run GLB services, you will need to be ready to configure your DNS to serve up all IP addresses in a round-robin fashion.

Stingray traffic managers can be deployed in each GLB location so that they can monitor local performance and availability, although this is not necessary. All monitoring can be performed remotely should you require your traffic managers in some alternative location. The Stingray units exchange monitoring information with their peers, regardless of their actual location, so that all traffic managers can work together to coordinate their operation.

Deployment planning

Where does Stingray fit?

Stingray contains a database of geographic locations for the public IP address space and acts as a DNS proxy. It can be deployed either in parallel with or in front of your current DNS infrastructure. When Stingray receives a DNS request, it forwards the request to a local DNS server. It then modifies the response that it sends back to the client based on a number of configurable metrics (including geographic location).

Deployment methods

DNS information is typically stored at a remote 'DNS registrar'. You can also store DNS information locally on your own authoritative DNS servers.

For this purpose, Stingray can be deployed in two ways:

- With an 'Inline Deployment', your traffic managers sit in front of the DNS servers that host the DNS records for domains to be managed. All DNS traffic is directed to Stingray (by modifying the NS record). Stingray then forwards it to the DNS servers and manipulates the response.
- With a 'Parallel Deployment', a new DNS subdomain is created, and the authoritative DNS server is configured to refer DNS requests to this new subdomain (using a CNAME record). The subdomain is hosted independently from the DNS registrar, and the Stingray traffic managers are the authoritative DNS servers for that domain.

The Parallel Deployment is slightly more complex to configure, but is a better solution if you are managing large numbers of domains or if you expect to make frequent changes.

Inline deployment

With an 'Inline Deployment', Stingray operates as the authoritative name server for each domain you serve. It forwards all DNS requests to the 'real' DNS server, and processes the responses.

The real DNS server may be the server at your registrar, or it may be a local caching DNS server.

You will need to edit two items of information that are stored at your DNS registrar:

- The NS record: This identifies the IP address(es) of the DNS servers that are authoritative for the domain. They typically point to DNS servers operated by the registrar. They should be changed to point to the IP addresses of your traffic managers.
- The A record: This identifies the IP address(es) that should be returned in a DNS lookup for the domain. All of the IP addresses of each location hosting the service should be returned. This configuration is known as 'Round-Robin DNS'.

Stingray then receives all DNS lookups for the domain to be load balanced. It forwards all requests to the real DNS server, and receives a list of IP addresses (in round-robin fashion) as a response. Stingray then processes this list to select precisely which IP address the client should receive.

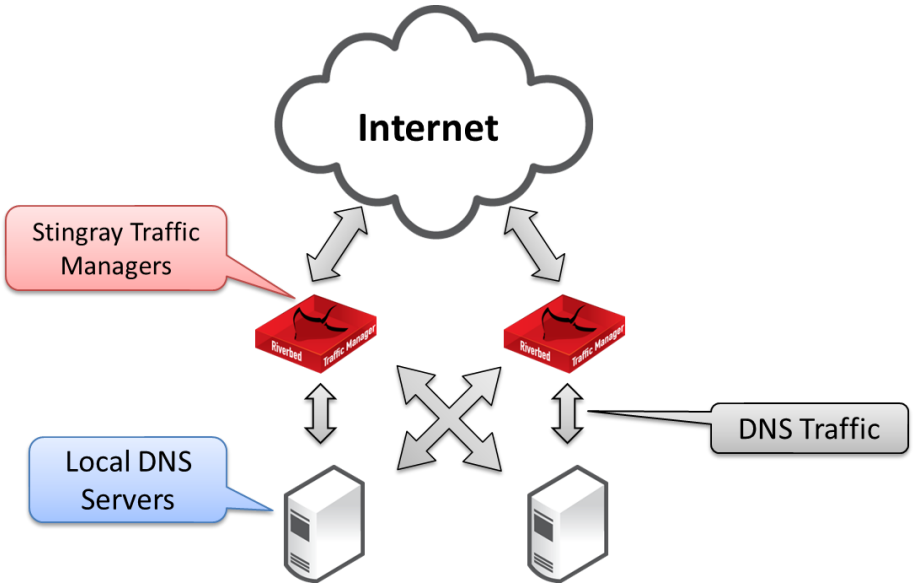


Fig. 47. Inline Deployment Diagram

Updating your Domain's Delegation Records

DNS settings on the 'real' DNS server will be controlled by your DNS provider and will likely be configured through a web interface.

When you log into your domain configuration page, somewhere you will see a list of name servers. Each of these will consist of a Fully Qualified Domain name and an IP address. To update these records, simply replace the IP Addresses and *Fully Qualified Domain Names* for your current DNS servers with those of your traffic managers. For further assistance with this, please contact your DNS provider.

Parallel deployment

With a parallel deployment, you configure a CNAME for your DNS entry that points to a separate domain that you have full control over. For example, suppose that your production site is accessed using the www.example.com domain name.

A CNAME would indicate that www.example.com is an alias for www.gslb.example.com:

- In the zone file for example.com hosted at your registrar, you add a CNAME record that aliases www.example.com to www.gslb.example.com. You also create an NS record for gslb.example.com, returning the IP addresses of your traffic managers.

You would then manage the entire configuration for the gslb.example.com domain:

- On your own internal DNS servers you host the gslb.example.com domain. The NS records for this domain point to your traffic managers, and the A record for the www.gslb.example.com domain name contains the IPs of each location hosting the www.example.com service (in round-robin fashion).

When a DNS client looks up the domain name www.example.com, it will contact the DNS server that is authoritative for the 'example.com' domain. This DNS server resides at your DNS registrar; it returns a CNAME response that effectively says 'try www.gslb.example.com instead'. The DNS client will then try to determine who is authoritative for the gslb.example.com domain. It will ask the authoritative DNS server for example.com for the NS record for gslb.example.com.

The DNS client will then query your traffic managers, believing them to be authoritative for gslb.example.com. It will send a request for www.gslb.example.com; Stingray forwards this to your internal DNS servers and receives the list of IPs for your locations. Stingray will process this list, returning back an A record containing one of the IPs of one of the locations.

This entire process is completely hidden from the end user.

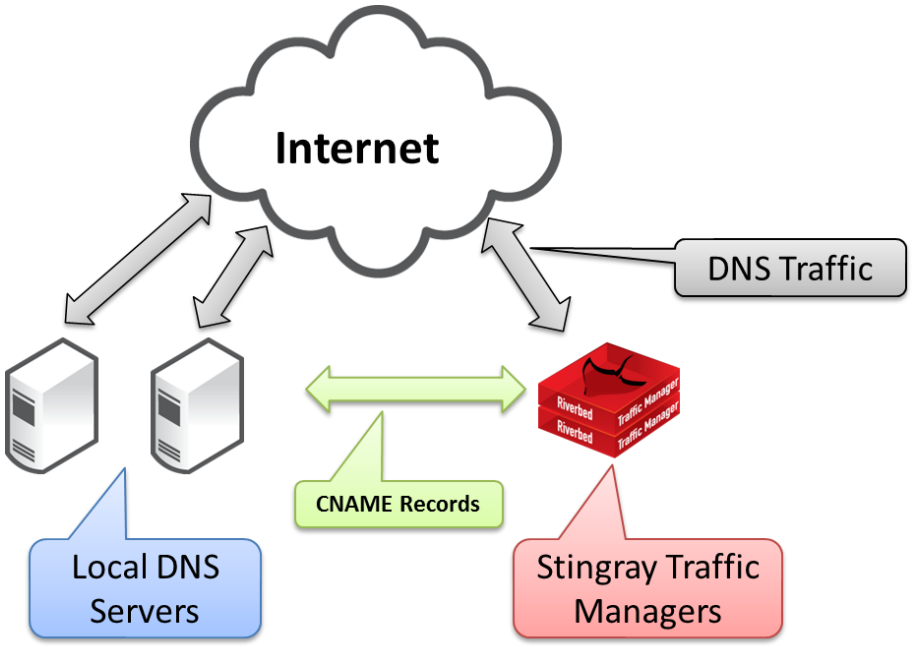


Fig. 48. Parallel Deployment Diagram

Parallel deployment is slightly more complex to deploy because you need to create a new DNS subdomain and host it on appropriate name servers. However, it is a more flexible technique, and is ideally suited if you expect to change your DNS configuration frequently, or wish to centralize the configuration for many domains as you can use the same subdomain for each.

For example, you can alias many different domain names (www.example.com, www.mysite.com and www.example.org) to the same CNAME (www.gslb.example.com) so that you can configure many domains centrally from a single CNAME.

The Time-to-Live (TTL) field

When you make a change to any of your DNS records, there is a good chance that the change will not have an immediate effect on your internet traffic. This is because each record contains a Time-to-Live (TTL) field. The TTL field essentially tells downstream DNS servers to cache this record for a given number of seconds; after which time it should send a new request. To ensure that your records propagate across the internet in a timely fashion, we recommend a TTL value between 30 and 60 seconds.

Configuration overview

This section will provide you with an overview of the Stingray GLB architecture and configuration settings. It outlines the key concepts of a working Global Load Balancer.

GLB locations

Global Load Balancer (GLB) locations are considered to be independent geographic service installations, sometimes referred to as datacenters. You may have two or more GLB locations in different countries around the world, deployed in order to provide a suitable application delivery infrastructure appropriate to local requirements, or for full disaster recovery for critical systems.

Within the context of Stingray, a GLB location is defined by its geographic position in the world, and is used by a GLB service to determine where DNS responses will direct clients.

GLB Locations are defined on the **Catalogs > Locations** page. Suitable values are a country name, a set of global coordinates, or simply a point on the supplied map.

Each location has its own *Local DNS Servers* (which Stingray will proxy DNS requests to), and its own *Health Monitors*. Both of these are configured on a service-by-service basis, and are covered in the *Editing a GLB service* section below.

Note: GLB locations are an independent concept to Configuration Locations, which are covered in the *Configuration Locations* section above. A Stingray traffic manager will itself be marked as present at a Configuration Location, yet this has no bearing on the GLB services it manages. Stingray traffic managers do not *need* to be physically present at a GLB location, however there are performance and availability benefits for being so.

GLB services

A GLB service is based on a set of Fully Qualified Domain Names and IP addresses. When you create a new GLB service, you tell Stingray which IP addresses are hosted at each GLB location for each domain name. All of the traffic managers manage DNS lookups for that service. See the *GLB services configuration* section below for full details on configuring GLB Services.

Virtual servers and pools

Each traffic manager will listen for incoming DNS requests through a specially-configured Virtual Server for your GLB services.

You can configure this virtual server through the **Manage a new service** wizard, or alternatively you can manually add a virtual server and pool through the **Services** section of the Admin UI.

With either method, select a protocol of **DNS (UDP)** from the drop-down list. Specify your DNS servers as the back-end nodes, providing a hostname/address and port for each one. Click **Finish** on the final page to add your new service.

Specifying the DNS (UDP) protocol enables a new **GLB Services** section on the **Virtual Servers > Edit** page for your chosen virtual server. Stingray will forward traffic to the GLB services you configure here. If you have multiple GLB services, Stingray will pick the correct one based on the domain being requested.

DNS servers

You may wish to install traffic managers in the same location as the DNS server(s) for performance reasons, but this does not have to be the case. However, all traffic managers in the same location will use the same set of DNS servers (whether these are local or remote).

If you specify more than one DNS server node for a particular location, the traffic managers will use each one according to the load balancing algorithm defined for that pool. DNS servers are defined as individual nodes in a DNS-specific pool. If one DNS server node times out, an alert will be raised and the traffic manager will try the next DNS server.

Traffic managers in different locations can use the same DNS servers. However, for performance and reliability reasons, it is best if your traffic managers are configured to use DNS servers local to them. This can achieved by combining **GLB Locations** with the **Configuration Location** functionality described in CHAPTER 28. Selecting the Location tool next to the node list on the pool edit page enables you to set DNS nodes on a per-location basis. DNS server nodes can then be added to or removed from each location as necessary. The overall location each traffic manager is set at will then determine which nodes are selected within the pool.

External connectivity monitors

If a location loses connectivity to the internet, Stingray will direct clients to a different location.

You should configure external connectivity monitors for a GLB location by creating a new **pool/GLB** monitor and monitoring services that are present in (or served by) that GLB location. If any monitor fails (within the failure threshold limits defined in the monitor), then the entire location is marked as 'down' and all of the traffic managers in the cluster will avoid that location when load-balancing clients.

The monitors catalog is described in detail in CHAPTER 14.

GLB services configuration

A GLB service represents the global load balancing configuration that you want to use for a set of DNS domains. Within a GLB service, you configure the following core information:

- The domains the GLB service applies to.
- The locations that are hosting the services for those domains.
- The load-balancing algorithm to use for the locations.

You can also configure additional settings:

- The TTL (Time-To-Live) value, which determines how frequently a DNS client checks the DNS information for the domain.
- Draining: you can drain a location to stop clients being directed to it.
- Monitors: you can configure one or more monitors to check the correct operation of the services and hosts in each location.
- Logging: you can log every DNS request and response for inspection and debugging.

Creating a new GLB location

Locations are listed and modified from the **Catalog > Locations** page. You can also find them through the **System > Traffic Managers** page, by clicking on the *Manage Locations* link. Your Configuration locations and GLB locations are listed separately on this page, with the opportunity to add to, and remove from, each list.

Clicking on the location name or corresponding **Edit** link allows you to view its settings page. Both types of locations have the following settings associated with them:

- **Name:** The identifying name for a location.
- **Position:** The geographic position for a location. This is only required for GLB locations, although can help to provide additional identifying

information for configuration locations. Select from either a drop-down list of country names, a specific set of latitude-longitude coordinates, or select a point on the displayed map.

Locations can be deleted from this page if they are no longer required. Note however that you must first disassociate your traffic managers with this location before removing it from the system.

Creating a new GLB service

Navigate to the **Catalogs > GLB Services** page, and enter the service details as requested in the *Create a new GLB service* section at the bottom. Required entries are service name, at least one domain name that you wish to balance globally, and one or more pre-defined locations that will host the service. If there are no locations shown, you must first create one or more GLB locations from the **Catalogs > Locations** page. Clicking on Create GLB Service will take you to the GLB service **Edit** page where further details and settings can be specified.

Editing a GLB service

The GLB service Edit page consists of a set of basic settings, as entered when the service was created, followed by a number of sub-sections that cover all other available configuration settings.

Basic Settings

Provide an alpha-numeric identifying name for your service that corresponds to the standard Stingray configuration naming convention.

The *enabled* switch allows you to in effect remove this service from operation without having to delete it and hence lose your configuration.

The *Domains* table allows you to add and remove domain entries that this service balances. Create an entry here for every *Fully Qualified Domain Name* that you wish to balance globally. You may use "*" as a wildcard. So, to manage all DNS lookups for any 'example.com' domain, provide the entry '*.example.com'.

Locations and Monitoring

In this section you can add and remove pre-defined GLB locations to your service. The *Add Location* section allows you to add new locations, and each added location contains a link to remove it. Within each added location, you can configure the following:

- **Draining Locations:** You may wish to completely stop sending traffic to a location. A practical need for doing so would be if you are

physically relocating your equipment from one location to another. Another example would be upgrading the backup power system and you need to disconnect your UPS, generator, inverter and/or batteries.

- **Local IPs:** In order for Stingray to manipulate the answers it sends to a client, it must know which IP addresses belong to which locations. Here you may enter a list or a range of IP addresses for each. These IP addresses correspond to the IP addresses that your DNS servers will return in a round-robin fashion when queried for the domains you are managing.

- **Monitors:** Here you can select which monitors should be used in this location. Each monitor performs one test, and multiple monitors may be configured for each location. Only those monitors which have a host/pool-wide scope can be selected. The monitor tests are performed at configured intervals against a specific machine. When a monitor has exceeded its limit of consecutive failures, the service in a particular data center will be flagged as dead; and it will be filtered out of any subsequent DNS responses. For information on creating a monitor, please refer to CHAPTER 14.

Load Balancing

There are several load balancing methods available. Note that the 'Disaster Recovery' variant of Stingray only supports the active-passive load-balancing method.

The load balancing methods are:

- **Load**: Distributes traffic based on the current load to each location. The decision of which location to send the client to is based purely on the load of each location; which it discovers via health monitors or the external SOAP API.

- **Geo**: Distributes traffic based solely on the geographic location of each client.

- **Adaptive**: Distribute traffic based on both the load and geographic location. If you find that one of your locations is receiving most of the

traffic (to the point where it is being overloaded) then you may wish to use this load balancing algorithm. This will help to achieve the best balance between latency due to distance and latency due to load.

- **Geo Effect**: You may fine tune how much effect the geographic location has when deciding which location a client should be sent to.

- **Active-Passive**: If you provide a service which is dependent on state information (such as a shopping cart site) and you are not replicating your databases globally in real time, you may wish to run your sites in an active-passive configuration. This algorithm will ensure that DNS records are only sent for one location at a time.

  - **Failover Order**

    If you enable Active-Passive mode, you will be able to choose which location is active and which ones are passive. If you have more than two locations, you may specify the order in which the locations fail over. The order is indicated from top-to-bottom. To promote a location, simply drag it to the desired place within the stack.

  - **Automatic Failback**

    You may also decide what happens when a higher-priority location recovers after a failure.

    If Automatic Failback is enabled, all users will be directed to the higher-priority location. It will become active.

    If Automatic Failback is not enabled, users will continue to be directed to the same location until you explicitly configure Stingray to reactivate the higher priority location using the setting in the user interface. This behavior is desirable in cases where the recovered location needs to synchronize state or content before it can be used.

DNS Authentication (DNSSEC)

The Domain Name System Security Extensions (DNSSEC) suite of specifications provide a set of security and authentication extensions to DNS. This section provides the facility to allow Stingray to alter DNSSEC authenticated responses by setting up associations between signature domains and DNSSEC Private Keys. You

can add one or more associations by entering a signature domain in the box provided and clicking the 'plus' symbol. The private key can then be selected from the drop-down box. Click Update to confirm the association.

You can associate multiple keys to one domain, and an individual key can be used to authenticate many domains. This is in essence a many-to-many relationship.

DNSSEC private keys can be added by clicking the *Manage DNSSEC private keys* link, which takes you to the **Catalogs > SSL > DNSSEC Keys** page. Please refer to CHAPTER 13 for more information on the importing/uploading of SSL Certificates and Keys.

For more information on DNSSEC, please refer to the website www.dnssec.net.

Rules

TrafficScript rules can be applied to this service to give sophisticated traffic management functionality. Note however that you cannot use functions that are incompatible with the DNS protocol (such as http.*). See CHAPTER 8 for full details of TrafficScript and its capabilities.

Request Logging

You may need more visibility of the DNS requests that are flowing into your Stingray traffic manager cluster. Request Logging lets you log each request into a specified file for later analysis. To enable logging, set the **log!enabled** config key to yes and click *Update*.

- **log!filename:** The location on the local file system where the request logs for this GLB Service should be located.
- **log!format:** The string format of the request log. Information about each request may be accessed via macros escaped by "%". The available macros are listed in the user interface.

Connection Settings

TTL (Time To Live): If you want to modify the TTL value in each response, you may do so here. The TTL value controls how long clients cache the DNS response before attempting to re-resolve it. The TTL values used by DNS servers are typically too long for GLB purposes (in the order of hours or days). To get a rapid failover if a location fails, clients need to re-resolve the DNS entry frequently, so a value of 30-60 seconds is appropriate.

Health Monitoring

Each individual GLB Service can be configured with a list of monitors that should be run in each GLB location. All of the monitors defined for a location must succeed for Stingray to consider that location in its load balancing decision for that service.

The monitors generally test a specific named machine for correct operation. For example, a monitor may test the primary server load balancer in that location to verify that it is correctly serving content. The primary server load balancer will have the IP address that is published using DNS for that service in that location.

You can assign multiple monitors to a location if you wish to test several machines for correct operation.

If you have configured multiple GLB Services, the monitors for each operate independently. Even though a location may fail for one GLB Service, the other GLB Services will continue to use that location if their monitors are successful.

Monitors can optionally provide performance information to the load-balancing algorithm, if you are using a 'Load' or 'Adaptive' load-balancing method. The load-balancing algorithm will then send more traffic to the location that reports the best performance.

Performance data is based on the time the monitor takes to complete, i.e. the response time of the monitored node.

Traffic Visualization

Introduction

When Global Load Balancing is enabled, Stingray provides additional information into the existing visualization and diagnostic tools in the Admin UI. For example, the World Map provides a real time view of incoming DNS requests and the geographic location to which the client is subsequently connected (see *The world map* section of CHAPTER 28 for more information).

The Current Activity Graph

The **Activity > Current Activity** page shows you a real-time graph of the activity on your system.

In the Settings section at the bottom, you can now choose to plot your data by *Traffic Manager* or *[Configuration] Location*. Both choices give you option to select all or some of the corresponding traffic managers/locations configured on your system, and whether to plot combined or separate results.

Two new sets of values have also been added to the monitor values tree on the **Current Activity > Edit** page containing a number of metrics pertaining to your GLB Services and Locations.

See CHAPTER 7 for more details on the Current Activity Graph.

The Historical Activity Graph

Stingray records the system's activity for the past 90 days. By clicking on the **Activity > Historical Activity**, you can view this data in graphical form. You can see the data plotted on either a linear or logarithmic axes.

In the **Settings** section at the bottom, you can now choose to plot your data by *Traffic Manager* or *[Configuration] Location*. Both choices give you option to select all or some of the corresponding traffic managers/locations configured on your system, and whether to plot aggregate or separate results. Timescales to plot range from 1 hour to 90 days.

See CHAPTER 7 for more details on the Historical Activity Graph.

The Connections Page

The Connections page gives you information about recent load-balancing decisions the GLB cluster has made; it is very useful in a test environment when investigating the behavior of your traffic managers. See CHAPTER 7 for more details on the Connections page.

Testing DNS with Dig

Dig is the recommended tool for testing and troubleshooting DNS systems. It is maintained by the Internet Software Consortium (ISC), the organization which also maintains Berkley Internet Name Domain (BIND). It is open source software and is available for many architectures and operating systems.

If you are a Linux user, depending on which distribution you run, dig is usually either installed with BIND or provided as supplemental package.

Visit the Internet Software Consortium's official website for further details:

http://www.isc.org/index.pl?/sw/bind

Example:

```
$ dig -t any +norec www.example.com @10.100.1.2
```

Where:

- **`-t any`** requests any type of record (A, NS, CNAME, PTR, etc).
- **`+norec`** indicates that the request should be sent with the recursive flag set to "off".
- **`www.example.com`** is the record we wish to test.
- **`@10.100.1.2`** is the name server that we are sending the request to.

When testing a DNS server or service, it is very important that you always perform non-recursive queries. This ensures that you will either get an authoritative answer or that you will know exactly which name servers are authoritative for the domain name you are trying to resolve. You will never receive an answer that the name server you are querying has looked up for you (since you can't always verify the source of that information).

CHAPTER 30 Troubleshooting

This chapter describes how to test that your traffic manager installation is working properly, and details the ways to deal with any problems you might encounter.

Tools and Techniques

If you are testing your traffic manager configuration, there are a range of tools and techniques you can make use of to understand your configuration, and the behavior of your network traffic:

Diagnosis: The **Cluster Diagnosis** page runs a number of tests across your traffic manager system, checking network connectivity, health monitors, configuration errors and conflicting configuration.

Event Log: The **Event Log** (displayed on the Home Page and the Event Log Page) records any significant events that have occurred.

Connections: The **Connections** report gives an instant display of ongoing and recent connections: where they came from, how they were handled, how much data was transferred, etc.

Request Logging: The traffic manager can write full logs so that you can record all the details of each request and response for later analysis.

Configuration Summary: If you have a sophisticated configuration, the configuration summary describes what configuration objects are in use, and how they are related.

Audit Log: The **Audit Log** records all configuration changes; when they were made, which user made them, and where the user came from.

This chapter describes each of these tools and techniques in detail.

Diagnosis and Event Logging

If you encounter any problems with your traffic manager system or configuration, the first step should be to check the **Cluster Diagnosis** page in the Stingray Admin Server.

Important: If you cannot access the Stingray Admin Server on any of the machines in your cluster, please try starting or restarting the traffic manager software as described in the **Installation and Getting Started Guide**. Take careful note of any error messages, particularly any messages appended to the error log file in `ZEUSHOME/zxtm/log/errors`.

Click the **Diagnose** button on any page of the Admin Server interface to view the current system status. The **Diagnose** page is divided into sections which will help you understand and resolve any issues you may have.

The **Cluster Diagnosis** section reports any problems the traffic manager finds with your setup, such as unavailable machines or bad configuration.

The **Event Log** shows the contents of the error logs on each traffic manager machine. You can view logs created in a recent period of your choice. The Event Log page dynamically updates as new event messages are added to the log.

The **Audit Log** shows recent configuration changes; when they were made, and which user made them.

The **Technical Support** section provides useful links to manuals, and an option to download technical support data for the Riverbed support team if necessary.

If a problem is detected during normal usage, the **Status Applet** will indicate a warning or error condition. Visit the **Cluster Diagnosis** page for full details of the problem.

Fig. 49. 'No errors' and 'Errors found' in the cluster warning screens

Monitoring Requests and Responses

If the **Cluster Diagnosis** page does not report any errors that are related to the problems you have observed, the next step is to verify that the traffic manager is correctly managing your traffic – reading requests on behalf of the back-end servers, and returning their responses back to the remote client.

You may find the suggestions in the *Generating test requests* section of CHAPTER 28 useful when attempting to reproduce any problems.

Connection Activity Report

Click the **Activity** button in the Stingray Admin Server and select the **Connections** tab. The **Connections** page gives a report of all ongoing and recently completed connections that the traffic manager has managed. Unfolding the **Key** section gives you details about the information shown.

You can see which traffic manager is currently handling the connection and the back-end node involved; the current state of the connection; and the virtual server, rule and pool currently involved with the connection. You can also see the retry count, idle times and byte counts for the connection.

To keep the data up to date, click the refresh button in your Internet browser. You can also download the data as a `.tsv` file.

Using the Connections Report

Note: Verify that the traffic manager has managed the requests you are testing. If you do not see your test requests in this list, it is probable that the traffic manager never received the requests. Check your client configuration and DNS settings to verify that your test client is sending the requests to the traffic manager IP address, and that the request is then managed by one of your back-end servers.

One common error is that the back-end servers issue self-referential responses. For example, suppose that the traffic manager is managing traffic to `www.example.com`, and the back-end servers are named `server1.example.com` and `server2.example.com`. One of the servers issues a response containing a link or redirect to `server1.example.com`. The client would subsequently try to contact the back-end server directly, bypassing the traffic manager and causing problems. Other protocols which embed DNS names or IP addresses in the response are also prone to this type of problem.

You can increase the size of the Connections report using the **System>Global Settings** page; look in the **Logging** section for the **recent_conns** key.

Request Logs

You can use the Request Logging facility of a virtual server to log many aspects of a transaction. This is very useful for long-term testing, if errors are intermittent or hard to replicate. Please refer to the *Request Logging* section of CHAPTER 4 for configuration instructions.

if you have complicated TrafficScript logic, you can store debugging information for the connection, and log it in the request log. Use the `connection.data.set()` function to store the specific data for the connection in a TrafficScript rule, and then log that data in the request log with the '%{Key-name}d' macro.

Advanced logging

You can use a TrafficScript rule to log information about the transactions managed by your virtual server.

TrafficScript functions can retrieve information about every part of a request. Functions such as `http.getHeader()` and `http.cookie()` give information about HTTP requests; functions such as `ssl.getSessionID()` provide details about SSL traffic. For other protocols, you can use the `connection.getData()` function to return a specified number of bytes of a request.

You can use the `log.info()` function to write details of the incoming requests to the traffic manager's event log file, to pinpoint any problems that are occurring with incoming traffic, or you can use the `event.emit()` function and configure an appropriate Event Handler to give you more control over where debugging messages are recorded.

See the **TrafficScript Manual** for details of the TrafficScript functions.

Important: You should take care when logging such information: if you do this for every incoming request, the log file will grow very rapidly.

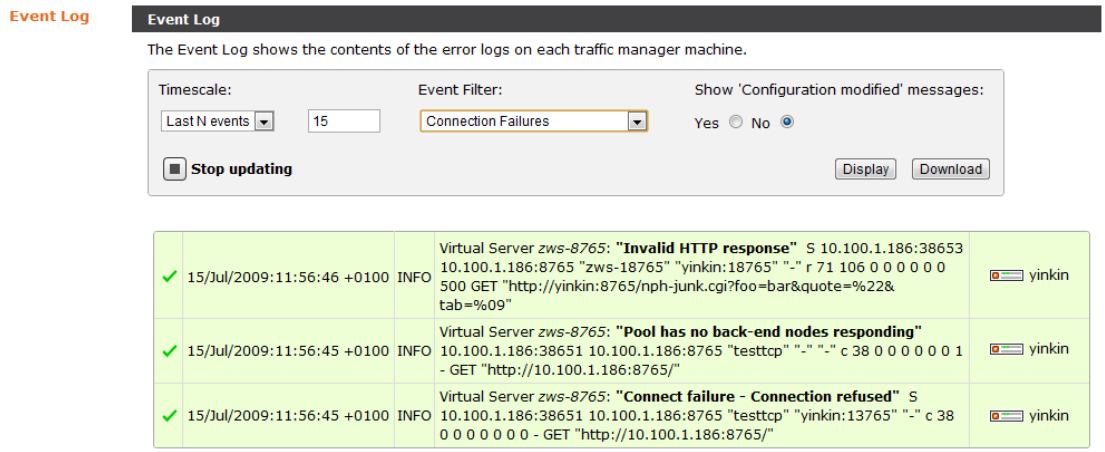
Monitoring Events

The TrafficScript `counter.increment()` function is used to count how many times a specific event occurred. For more information, please refer to the *Activity > Current Activity* section of CHAPTER 7.

Detailed debugging of connections

If a connection fails, the information logged in the connection activity report and request log will be incomplete because the connection did not complete normally. Connections may fail because of a variety of reasons, including protocol errors, timeouts or unexpected TCP closes.

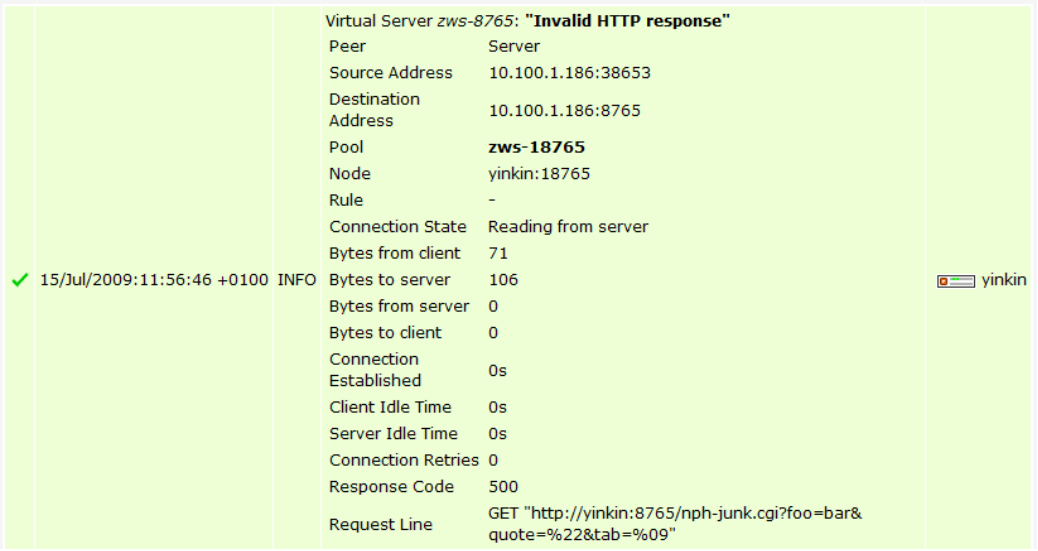
To debug such unexpected problems, enable the **log!client_connection_failures** and **log!server_connection_failures** settings in the **Virtual Server > Connection Management** configuration page for the affected service, as described in the *Handling Errors* section of CHAPTER 4. These settings will configure your traffic manager to write detailed debug messages to the Event Log whenever a connection fails:



You can filter the event log to only display these messages using the 'Connection Failures' Event Filter as illustrated above.

The log line contains a description of the error (for example, "Invalid HTTP Response") and details that describe the nature of the connection (node used, connection times, retries, etc). The online help for the **Virtual Server > Connection Management** configuration page describes the log format in detail.

Click on the description to unfold a more readable representation of the connection error:



By default, connection errors are logged to the global Event Log. You can create an Event Handler that captures all Connection Failure events and writes them to a separate log file, and bypasses the global Event Log. Refer to the *Overview* section of CHAPTER 21 for more details.

Testing individual nodes

To check that each back-end node in a pool can receive requests from the traffic manager, switch the load-balancing algorithm for the pool to **Round Robin** (see the *Load Balancing* section of CHAPTER 5). The 'Round Robin' load-balancing algorithm routes requests to each node in turn; you can use the logs on your back-end servers to check that this is working for every node.

You can also configure nodes to drain connections (see the *Draining and Disabling Nodes* section of CHAPTER 5). If you have several nodes in a pool, you can temporarily stop the traffic manager sending traffic to some by setting them to drain. To test each node in turn, you can drain all the nodes apart from the one you wish to test.

Enable server logging on each node and verify that the logs the nodes write are consistent with the activity on the system.

Understanding Your Configuration

The configuration summary provides an overview of how requests are processed with the traffic manager. Requests are received by the traffic manager, processed by a virtual server and a pool, forwarded to a node, and then the response is processed by the pool and virtual server on the return path.

The processing path can be quite complex when a number of TrafficScript rules and the traffic manager capabilities are in effect.

The Configuration Summary is described in section CHAPTER 7.

If you can trace the problem under investigation to a particular time, you can use the **Audit Log** (in the **Diagnose** part of the Admin Server) and possibly the **Backup Management** (in the **System** part of the Admin Server) to identify configuration changes that may have provoked the problem.

The Audit Log records all configuration changes performed by each user of the Stingray Admin Server interface. You may wish to give each authorized administrator an individual login so that you can manage their privileges and monitor their activities. User and Group configuration is described in the *User Management* section of CHAPTER 24.

Backup Management allows you to manually take snapshots of your configuration. You can compare the current configuration with a known good snapshot, and restore backups in the event of a serious configuration problem. Backup Management is described in the *System > Backup* section of CHAPTER 7.

Troubleshooting Tips

The process of diagnosing faults in a complex traffic-managed cluster is an involved one, and a systematic approach to troubleshooting is required. In the event of any problems, check through the following areas in turn to locate and diagnose the cause of the fault.

Generating test requests

You can test your system using the normal client software that users will employ, such as a web browser or email client. Tools like the Live HTTP Headers extension for Mozilla browsers are very useful when inspecting the request and response flow between the client and the traffic manager system.

You can also use a network snooping tool such as Wireshark (http://www.wireshark.org/) to record network traffic and assemble request and response sessions.

httpclient

The `httpclient` program, included with the traffic manager distribution, can be used to issue HTTP requests to particular machines. You can find it at *ZEUSHOME*`/admin/bin/httpclient`.

Use the following syntax:

```
$ httpclient --hostheader=<website_name>
http://<trafficIP>/
```

You can also perform initial tests using a telnet client to perform a basic test on the service:

```
$ telnet www.mysite.com 80
Trying 62.254.209.66...
Connected to 62.254.209.66.
GET / HTTP/1.1<RETURN>
Host: www.mysite.com<RETURN>
<RETURN>
```

The openssl toolkit (http://www.openssl.org/) includes a telnet-like client that uses the SSL protocol, for testing SSL-related problems.

zeusbench

The `zeusbench` program is a useful benchmarking tool that can be used to send large numbers of HTTP requests for load and performance testing purposes. You can find it at *`ZEUSHOME`*`/admin/bin/zeusbench`.

There are many command line options. You can run a simple load test using the following command:

```
$ zeusbench -t 30 -c 100 -k http://host/url
```

Run `zeusbench -h` for a full list of command line options.

Checking Automatic Back-End Fail-Over

To check the traffic manager's automatic fail-over of back-ends you will need at least two back-end servers configured, or there will be no machines for the traffic manager to fall back on. You can test fail-over by pulling out the network cable on one of the back-end machines; or you can manually stop the service running on the back-end. Verify that nothing is then listening on that port on the back-end.

If you only have one back-end server you could run two instances of the required service on different ports on the same server machine, and then manually stop one instance of the service.

Try to use your selected service through the traffic manager. For SMTP send an email through the server farm, or for HTTP make a web page request. If at least one back-end server is available to fulfil your request it should succeed.

Check the event log, linked from the **Diagnose** pages, for notification that a back-end machine has failed.

Checking Automatic Front-End Fail-Over

If you have two or more traffic managers, you can check that automatic front-end fail-over is working properly. Set up a traffic IP group spanning your traffic managers, and a service using this traffic IP group, such as a virtual server managing web content on port 80. Check that you can request web pages from this service successfully on each of the traffic IP addresses in the group. You can do this by entering the IP address rather than the DNS name in your browser.

Now click the **Services** button on the Admin Server pages. Click the **Traffic IP Groups** tab, and click **Unfold All** to view details of your traffic IP groups. This shows you which machine has raised which IP address; note that if you have more machines than traffic IPs in the group, some machines will be on standby and not actively handling traffic.

Now pick a machine that has raised one of the traffic IPs, and pull out all the network cables on that machine. The IP address will be raised by another machine in the group; try browsing to the traffic IP address again and check that you can still receive content. You may need to refresh the page in your browser to ensure that it is not using cached content.

The *Understanding a traffic manager's fault tolerance* section of CHAPTER 6 describes how Stingray's traffic management fault tolerance works, the tests that it conducts and the decisions that it makes.

Common Problems

Did not become root

The traffic manager needs to bind to privileged ports to balance network services like mail and HTTP (privileged ports are ports below number 1024). To do this you need to install, run the configure script and start the traffic manager as root.

In front-end fault tolerance mode, the traffic manager needs to raise and lower network interfaces. If you did not configure and start the software as root, the fail-over architecture will not function.

Note that the traffic manager runs as a non-privileged user, but it has to be configured as, and started as the root user.

For evaluation purposes, it is possible to configure and start the traffic manager without becoming root. However, in this case you will not be able to bind to privileged ports (see above), nor use fault tolerance.

Connection refused

The most common configuration error for front-ends is a bad DNS setup mapping the external name to the external IP addresses for your front-end machines. If you receive a "Connection Refused" message when connecting to your server through the front-end DNS name, it is likely that your DNS is not configured correctly.

To check your DNS configuration, you can use the `host` or `nslookup` commands:

```
$ host www.w3.org
www.w3.org has address 18.29.1.35
www.w3.org has address 18.7.14.127
www.w3.org has address 18.29.1.34
```

Verify that your traffic manager machines are listening on these IP addresses.

You can configure a virtual server to listen for traffic on these IP addresses explicitly. Then the Diagnose page will report an error if these IP addresses are not available to the traffic manager cluster.

Inappropriate Traffic IP Addresses configured

If you configure a traffic IP group for front-end fault tolerance, the traffic manager will make a number of checks to ensure that the traffic IPs you select are sensible. However, you should also double-check that they are appropriate for your current network configuration. Common errors include selecting traffic IPs that are used elsewhere (including as permanent IPs on the traffic manager machines), IPs that do not lie in the subnets used by the traffic managers, or IPs that are not routable from elsewhere.

The traffic manager drops connection before protocol begins

For *server first* protocols, when the traffic manager cannot contact any back-ends for work for a particular port for server first protocols, it will have no choice other than to drop the connection with the client. If you connect to the correct port on the traffic manager machine (using telnet for instance), and get a successful connection (i.e. no "Connection Refused" message), but the connection drops almost immediately, the traffic manager is trying to talk to a back-end but cannot find any.

Check the event log on the Diagnose page for messages about the status of the back-end servers.

Web Server returns Error 400

If on testing your traffic-managed website you only get errors of type Error 400 Bad Request, this may be because the website you are trying to access has not been configured to accept any other host headers (or aliases). In order to resolve this problem you can add a rule to set the right host header:

```
http.setHeader( "Host", "www.mysite.com" );
```

Wrong port number configured

Another common problem is the wrong port number being balanced. If, for instance, your website is running on a port other than 80, you will need to balance that port number with the HTTP protocol rather than the default. If you are running an SSL-encrypted site, you should select the HTTPS protocol rather than HTTP, or SSL-decrypt your traffic. The default port for HTTPS is 443.

Note that it is possible to balance many distinct ports with one traffic manager installation, provided an appropriate protocol is selected in each case.

Running out of file descriptors

Your traffic manager uses operating system resources called 'file descriptors' to manage each network connection to a client or server. There is a hard limit on the number of file descriptors it can allocate (the limit applies per process, i.e. per CPU core), set in the operating system.

Under times of very high load, such as during a benchmark, you may see an error message indicating that you are "Running out of file descriptors". In this case, you can increase the number of file descriptors that your traffic manager uses.

Go to the **System > Global Settings** page in the administration interface. In the **System Settings** part of the configuration, increase the value of **maxfds** to a larger value. If you are already at the maximum that your operating system allows, you should refer to your system documentation to ascertain whether this hard limit can be increased.

Getting Help

If you need to contact a technical support engineer, please include the **Technical Support Report** download as an attachment to your problem report. The report download contains your recent log files, configuration, system activity and a variety of internal information, and can be downloaded from the **Technical Support** page in the **Diagnose** section of the Admin Server.

A **Technical Support Report** can also be generated from the command line. This ability is provided to cover for occasions when access through the UI is either (a) not possible or (b) too slow.

The command line Support Report generator also allows for the report to be stored to any location on the traffic manager's hard drive. This allows for the storage of the report to a different partition if the root partition (/) has become full for some reason.

Run the following command from the traffic manager system command line:

```
# ZEUSHOME/zxtm/bin/support-report <output file>
```

CHAPTER 31 Further Resources

Stingray Manuals

Your traffic management system includes an **Installation and Getting Started Guide**, intended to get you up and running quickly. There are also full manuals for the TrafficScript rules language, Java Extensions and the Stingray Control API.

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/docs/stingray/index.htm

Online Help

Click the **Help** button on any page of the Admin Server interface to see detailed help information for that page. You can also view contents and index pages to navigate around the online help.

You can access any of the manuals by clicking the **Manuals** button on any of the **Help** pages.

The **Rules > Edit** page also has a link to **TrafficScript Help**, a quick reference guide to the functions.

Stingray information online

Product specifications can be found at:

http://www.riverbed.com/us/products/stingray/

Visit the Stingray Community website or Technical Blog for further documentation, examples, white papers and other resources:

http://community.riverbed.com

http://blogs.riverbed.com/stingray/

CHAPTER 32 Glossary

Below is an explanation of terms used in this document. Some of these terms are used with varying meanings in computing literature; this glossary is intended only to define them within the scope of this document.

Action – An action is invoked when an event handler is triggered as the result of a particular event occurring. Possible actions include writing to log files, sending an e-mail message, sending a SYSLOG message or SNMP trap, or running a custom action program.

Admin Server – The administration interface to the Stingray traffic management family of solutions, accessed through the web-based user-interface.

Audit log – The traffic manager's audit log of users' activities for the traffic manager. These can be viewed within the Diagnose pages.

Back-end IP address – The IP address which a traffic manager machine uses to communicate with a back-end server, or the IP address of a back-end server.

Back-end server – A machine which runs a service, such as a web or mail server. A back-end server is typically within your local network, and requests handled by the traffic manager are passed on to it. A back-end server together with a specified port forms a node in the traffic manager.

Bandwidth management – The ability to monitor and control the amount of network bandwidth available to a particular service or type of request.

Catalog – A central repository for objects which you can apply to your services. There are catalogs for rules, monitors, service protection classes and various SSL items.

Certificate authority – A recognized authority which independently signs SSL certificates.

Certificate revocation list (CRL) – A list of client SSL certificates which should be considered invalid; issued by a certificate authority.

Certificate signing request (CSR) – A request created from a self-signed certificate. You can submit the CSR to a certificate authority for them to sign.

CIDR IP subnet – A Classless Inter-Domain Routing (CIDR) IP subnet includes a standard 32-bit IP address and additional information on how many bits are used for the network prefix. For example, in the CIDR IP subnet "10.13.1.48/22", the "/22" indicates that the first 22 bits are used to identify the unique network prefix and the remaining 10 bits identify the specific host. Alternatively, the traffic manager allows to specify a CIDR IP subnet using a subnet mask "10.13.1.48/255.255.255.0" or explicitly "10.13.1.".

Client certificate – An SSL certificate held by a client, granting the client access to a restricted site.

Cluster – A group of traffic managers which share the same configuration.

configure script – The script which must be run after installing and before starting the traffic manager. It deals with fundamental settings such as passwords and specifying whether the traffic manager should stand alone or join an existing cluster.

Connection management – Settings for managing the connection between a remote client and a virtual server, or between a pool and its nodes.

Cookie - A small item of data given to a client by a server that is stored either on the client's file system or in the browser client session. The client stores the data and provides it to the server on subsequent requests. Cookies are used to track client data such as session persistence maps.

Denial of Service attack – A malicious attempt to prevent legitimate use of a service, often by flooding a network or disrupting connectivity between machines.

Diagnose pages – The section of the Admin Server which helps you diagnose and fix any faults with your traffic manager setup.

Disabled node – A node in a pool that is not used; it is not monitored, and no traffic is sent to it. Disabling a node is an alternative to removing the node from the pool, but disabling it allows it to be reinstated more easily.

Distributed Denial of Service attack – A Denial of Service attack which comes from a group of machines; these machines have often been compromised by worms or viruses.

DMZ – The demilitarized zone for a firewall. This zone sits between the internal network and the Internet, and often holds your externally available servers.

Draining node – A node in a pool which is being sent no new connections. When all existing connections and sessions to this node have expired, it can be removed from the pool safely.

Error file – A file sent to the client when no nodes are available to respond to its request. Error files are set up on a per-pool basis.

Event – An event is raised when a particular change or occurrence takes place, such as an IP Address transfer or a node failure or recovery. All events are logged to the global Event Log.

Event Handler – An event hander configures the actions that the traffic manager should take when an event in the associated Event Type occurs.

Event Log – The traffic manager's log of events within the traffic manager. These are graded as INFO, WARN, SERIOUS, FATAL etc, and can be viewed within the Diagnose pages.

Event Type – A set of events, grouped for administrative convenience. When any event in a particular event type occurs, the traffic manager can run the actions associated with that event type.

Fail-over – The act of taking on a failed machine's load by other machines in the cluster.

Failure pool – If all the nodes in a pool should fail, requests will be sent to its failure pool, if one is configured.

Fault tolerance – The ability of a cluster or pool to cope with failure of one or more of its servers, without interruption to service.

Front-end IP address – The permanent, externally available IP address of a traffic manager.

Front-end server – A server which is contactable from the Internet, such as a traffic manager.

Health monitoring – The process by which the traffic manager tracks the health of traffic managers and back-ends, so that requests will not be sent to a failed machine.

IPv6 - A network layer protocol that is expected to supersede the current protocol, IPv4. Its features include a bigger range of addresses, higher security and further advantages.

Java extension - Java Extensions can increase ('extend') the capabilities of TrafficScript. A TrafficScript rule (or RuleBuilder rule) can invoke a Java Extension to process a request or response, and the Java run-time environment provides a much more powerful set of programming language tools and libraries than TrafficScript does alone.

Load balancing – The distribution of requests among a number of back-end servers, so that every request is dealt with as quickly as possible.

MD5 - In cryptography, MD5 (Message-Digest algorithm 5) is a widely-used cryptographic hash function with a 128-bit hash value. As an Internet standard (RFC 1321), MD5 has been employed in a wide variety of security applications, and is also commonly used to check the integrity of files.

MIME type - Multipart Internet Mail Extension, a name which identifies the type of data being transferred, e.g. 'application/pdf' for PDF files. This standard was first used in email applications, and has been widely adopted by other internet technologies such as web servers.

Monitor – A regular test which the traffic manager uses to track the health of a pool.

Name server – A server that implements a name service protocol. For example, a Domain Name System (DNS) server might translate the domain name www.riverbed.com to the IP address 208.70.196.59.

NAT – Network address translation (NAT, also known as network masquerading) involves rewriting the source or destination addresses of IP packets as they pass through a router or firewall. Most systems using NAT do so in order to enable multiple hosts on a private network to access the Internet using a single public IP address.

Node – A back-end server with an associated port, which receives requests sent to it by the traffic manager and responds with the requested content.

PEM –Private Enhanced Mail. The .pem file extension is used for Base 64-encoded X.509 keys and certificates. These are printable text, and start with a line of the form: ----- BEGIN CERTIFICATE -----

PKCS#11 – The standard used to communicate with secure hardware.

Pool – A logical group of nodes. When a virtual server assigns requests to a pool, it load-balances them across the nodes.

Real Time Streaming Protocol (RTSP) - An application-level protocol that enables on-demand delivery of real-time data, such as audio and video formats, using the TCP and UDP protocols.

Real-time Transport Protocol (RTP) – A UDP packet format for delivering audio and video files, which is commonly used alongside RTSP to provide the delivery of the streaming data.

Request Rule – A TrafficScript or RuleBuilder rule that controls how the traffic manager should handle a request. Rules are invoked by a Virtual Server, and can inspect, manipulate, rewrite and route requests; they can also control how other capabilities in the traffic manager are used to manage the request.

Response Rule – the counterpart of request rules, response rules control how the traffic manager should handle a response.

RuleBuilder - A visual editor for creating rules, which allows you to select conditions on the request to be examined, and actions that follow if any or all of these conditions are met.

Self-signed SSL certificate – An SSL certificate signed by its owner, rather than a certificate authority. These are useful for security testing but should not be used for live sites. You can issue a certificate signing request to convert your certificate to one signed by a certificate authority.

Server certificate – An SSL certificate used to identify an SSL service (such as a web site), and containing the public key necessary to set up the encrypted transaction.

Service – Common Internet services include web, mail, DNS and applications such as .NET and SOAP. They are provided, publicly or locally, via an IP or DNS address and port combination.

Service level monitoring – The ability to monitor back-end server response times, compare them to a conformance value and react to

changes, including dynamically adjusting the resources available.

Service protection – The protection of your Internet services from attacks such as denial of service (DoS) and distributed denial of service (DDoS).

Service protection class – A group of service protection settings you define. The class is held in the Service Protection Catalog and can be applied to any virtual server.

Session Initiation Protocol (SIP) - A protocol used in Internet for conferencing, telephony, presence, events and instant messaging.

Session persistence – A process by which requests belonging to the same user session are always sent to the same node. Sessions can be identified by several methods including IP addresses, cookies or SSL session IDs.

SNMP – Simple Network Management Protocol, an open standard for network monitoring and management. It allows you to monitor devices on a network and gather performance data.

SOAP - Simple Object Access Protocol, an XML-based standard for Web services messages.

SSL – Secure Sockets Layer, a protocol which provides encrypted communications over a network. Requests can be decrypted and re-encrypted within the traffic manager.

Stingray – The group name for the range of application delivery controller/traffic management products and variants supplied by Riverbed.

Tarball –A file in the tar file format, a standard type of archive file format. It is used widely to archive and unarchive files while preserving file system information such as user and group permissions, dates, and directory structures.

Traffic IP address – An IP address you use to publish an Internet service, usually linked to your public DNS entry.

Traffic IP group – A logical group containing several traffic IP addresses, and spanning some or all of your traffic managers. These traffic managers negotiate to ensure that the IP addresses in the group are always available.

Traffic manager – A front-end server running the traffic manager and managing your services.

Traffic cluster – The ability to deploy unlimited numbers of active and passive traffic managers, providing resilience to multiple failures and the ability to scale the traffic manager cluster as the need arises. Also known as N+M Redundancy.

TrafficScript – The scripting language used to write rules for the Stingray traffic management range of solutions.

Vector – The means or vehicle by which a Denial of Service or Distributed Denial of Service attack is made.

Virtual server – An interface between the traffic manager and the Internet. A virtual server manages one of your services. It receives a request from the Internet and, possibly after SSL-decrypting it and applying rules, assigns it to a pool.

XML – Extensible Markup Language, used for storing and exchanging structured data.

XPath – A language for addressing parts of an XML document. It can be used in conjunction with TrafficScript rules to make decisions based on the document's content.

ZEUSHOME – The location in which you install the traffic manager. The default is /usr/local/zeus for Stingray software variants, and /opt/zeus for Stingray virtual appliances/cloud instances.

CHAPTER 33 Software License Acknowledgements

License for the Berkeley DB code (version 1.85)

The following license refers to the libdb code that the traffic manager is linked against. The source for this code can be found at:

http://www.sleepycat.com/update/snapshot/db.1.85.tar.gz

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

All advertising materials mentioning features or use of this software must display the following acknowledgement:

This product includes software developed by the University of California, Berkeley and its contributors.

Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

RSA PKCS11

License to copy and use this software is granted provided that it is identified as "RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki)" in all material mentioning or referencing this software or this function.

License is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki)" in all material mentioning or referencing the derived work.

License for the OpenLDAP code, version 2.4.23

The following license refers to the libldap code that the traffic manager is linked against. The source for this code can be found at:

ftp://ftp.openldap.org/pub/OpenLDAP/openldap-stable/openldap-stable-20100719.tgz

The OpenLDAP Public License

Version 2.8, 17 August 2003

Redistribution and use of this software and associated documentation ("Software"), with or without modification, are permitted provided that the following conditions are met:

1. Redistributions in source form must retain copyright statements and notices,

2. Redistributions in binary form must reproduce applicable copyright statements and notices, this list of conditions, and the following disclaimer in the documentation and/or other materials provided with the distribution, and

3. Redistributions must contain a verbatim copy of this document.

The OpenLDAP Foundation may revise this license from time to time. Each revision is distinguished by a version number. You may use this Software under terms of this license revision or under the terms of any subsequent revision of the license.

THIS SOFTWARE IS PROVIDED BY THE OPENLDAP FOUNDATION AND ITS CONTRIBUTORS “AS IS” AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OPENLDAP FOUNDATION, ITS CONTRIBUTORS, OR THE AUTHOR(S) OR OWNER(S) OF THE SOFTWARE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,

PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The names of the authors and copyright holders must not be used in advertising or otherwise to promote the sale, use or other dealing in this Software without specific, written prior permission. Title to copyright in this Software shall at all times remain with copyright holders.

OpenLDAP is a registered trademark of the OpenLDAP Foundation.

Copyright 1999-2003 The OpenLDAP Foundation, Redwood City, California, USA. All Rights Reserved. Permission to copy and distribute verbatim copies of this document is granted.

License for the Cavium software

CN1000 and libZcn1000

The CN1000 driver, libZcn1000.so includes software developed by Cavium Networks, to which the following copyright notice applies:

Copyright (c) 2003-2005 Cavium Networks (support@cavium.com). All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice,this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by Cavium Networks

4. Cavium Networks' name may not be used to endorse or promote products derived from this software without specific prior written permission.

User agrees to enable and utilize only the features and performance purchased on the target hardware.

This Software, including technical data, may be subject to U.S. export control laws, including the U.S. Export Administration Act and its associated regulations, and may be subject to export or import regulations in other countries. You warrant that You will comply strictly in all respects with all such regulations and acknowledge that you have the responsibility to obtain licenses to export, re-export or import the Software.

TO THE MAXIMUM EXTENT PERMITTED BY LAW, THE SOFTWARE IS PROVIDED "AS IS" AND WITH ALL FAULTS AND CAVIUM MAKES NO PROMISES, REPRESENTATIONS OR WARRANTIES, EITHER EXPRESS,IMPLIED,STATUTORY, OR OTHERWISE, WITH RESPECT TO THE SOFTWARE, INCLUDING ITS CONDITION,ITS CONFORMITY TO ANY REPRESENTATION OR DESCRIPTION, OR THE EXISTENCE OF ANY LATENT OR PATENT DEFECTS, AND CAVIUM SPECIFICALLY DISCLAIMS ALL IMPLIED (IF ANY) WARRANTIES OF TITLE, MERCHANTABILITY, NONINFRINGEMENT,FITNESS FOR A PARTICULAR PURPOSE,LACK OF VIRUSES, ACCURACY OR COMPLETENESS, QUIET ENJOYMENT, QUIET POSSESSION OR CORRESPONDENCE TO DESCRIPTION. THE ENTIRE RISK ARISING OUT OF USE OR PERFORMANCE OF THE SOFTWARE LIES WITH YOU.

CN2000 driver and libZcn2000

The CN2000 driver, libZcn2000.so includes software developed by Cavium Networks, to which the following copyright notice applies:

Copyright (c) 2003-2005 Cavium Networks (support@cavium.com). All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. All manuals,brochures,user guides mentioning features or use of this software must display the following acknowledgement:

This product includes software developed by Cavium Networks

4. Cavium Networks' name may not be used to endorse or promote products derived from this software without specific prior written permission.

5. User agrees to enable and utilize only the features and performance purchased on the target hardware.

This Software, including technical data, may be subject to U.S. export control laws, including the U.S. Export Administration Act and its associated regulations, and may be subject to export or import regulations in other countries. You warrant that You will comply strictly in all respects with all such regulations and acknowledge that you have the responsibility to obtain licenses to export, re-export or import the Software.

TO THE MAXIMUM EXTENT PERMITTED BY LAW, THE SOFTWARE IS PROVIDED "AS IS" AND WITH ALL FAULTS AND CAVIUM MAKES NO PROMISES, REPRESENTATIONS OR WARRANTIES, EITHER EXPRESS,IMPLIED,STATUTORY, OR OTHERWISE, WITH RESPECT TO THE SOFTWARE, INCLUDING ITS CONDITION,ITS CONFORMITY TO ANY REPRESENTATION OR DESCRIPTION, OR THE EXISTENCE OF ANY LATENT OR PATENT DEFECTS, AND CAVIUM SPECIFICALLY DISCLAIMS ALL IMPLIED (IF ANY) WARRANTIES OF TITLE, MERCHANTABILITY, NONINFRINGEMENT,FITNESS FOR A PARTICULAR PURPOSE,LACK OF VIRUSES, ACCURACY OR COMPLETENESS, QUIET ENJOYMENT, QUIET POSSESSION OR CORRESPONDENCE TO DESCRIPTION. THE ENTIRE RISK ARISING OUT OF USE OR PERFORMANCE OF THE SOFTWARE LIES WITH YOU.

PCRE License

The following license refers to the pcre code that the traffic manager is linked against .

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of the University of Cambridge nor the name of Google Inc. nor the names of their contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Libnet License

The following license refers to the libnet code that the traffic manager is linked against (on Solaris and FreeBSD versions).

libnet 1.1.x

Copyright (c) 1998 - 2002 Mike D. Schiffman <mike@infonexus.com>

http://www.packetfactory.net/libnet

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

License for libexecinfo

Copyright (c) 2003 Maxim Sobolev <sobomax@FreeBSD.org>

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED ARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

$Id: execinfo.c,v 1.3 2004/07/19 05:21:09 sobomax Exp $

License for Yahoo! UI Library

Copyright (c) 2009, Yahoo! Inc. All rights reserved.

Redistribution and use of this software in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of Yahoo! Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission of Yahoo! Inc.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,

SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

License for ssleay cryptographic library

The following license refers to an implementation of the SHA-1 and MD5 algorithms used in Stingray Traffic Manager:

Copyright (C) 1997 Eric Young (eay@cryptsoft.com). All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- All advertising materials mentioning features or use of this software must display the following acknowledgement: "This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)". The word 'cryptographic' can be left out if the routines from the library being used are not cryptographic related :-).

- If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgement: "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS

FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

License for libxml2 and libxslt

The following license refers to the XML parser and toolkit libraries used in Stingray Traffic Manager, as licensed under the MIT License:

libxml2 2.7.5
libxslt 1.1.24

Copyright (c) 1999-2006 Daniel Veillard

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,

OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.

License for the Java Servlet API

The Java Servlet API used in Stingray Traffic Manager contains code from the Apache Jakarta Project, and is licensed under the ASF license version 2.0:

Copyright 1999-2007, The Apache Software Foundation

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

License for the expat XML parser

The following license refers to the expat XML parser library used in Stingray Traffic Manager:

expat 2.0.0

Copyright (c) 1998, 1999, 2000 Thai Open Source Software Center Ltd
and Clark Cooper
Copyright (c) 2001, 2002, 2003, 2004, 2005, 2006 Expat maintainers.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included
in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE
FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF
CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH
THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

License for MooTools

Stingray Traffic Manager contains code from the MooTools Javascript framework, licensed under the open source MIT license:

MooTools 1.2

Copyright (c) 2006-2009 Valerio Proietti

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO
EVENT SHALL THE

AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Licenses for OpenLayers

Stingray Traffic Manager contains code from MetaCarta, Rico and XMLHttpRequest.js. The following copyright notices apply:

Copyright (c) 2005-2008 MetaCarta, Inc.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted (subject to the limitations in the disclaimer below) provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- Neither the name of MetaCarta, Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

NO EXPRESS OR IMPLIED LICENSES TO ANY PARTY'S PATENT RIGHTS ARE GRANTED BY THIS LICENSE. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT

NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Contains portions of Rico <http://openrico.org/>
Copyright 2005 Sabre Airline Solutions
Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at:
http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Contains XMLHttpRequest.js <http://code.google.com/p/xmlhttprequest/>
Copyright 2007 Sergey Ilinsky (http://www.ilinsky.com)
Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at:

http://www.apache.org/licenses/LICENSE-2.0

License for rsync

Stingray Traffic Manager contains code from rsync – a fast remote file copy program. The following copyright notice applies:

Version: 2.6.9

License summary: GPL v2

Copyright:

rsync was originally written by Andrew Tridgell and has been improved by many developers around the world. rsync may be used, modified and redistributed only under the terms of the GNU General Public License, found at:

http://www.fsf.org/licenses/old-licenses/gpl-2.0.html

License for mod_imap.c

Stingray Traffic Manager contains code from mod_imap.c, which originates from the Apache HTTP Server (http://httpd.apache.org) and is licensed under version 1.1 of the Apache Software License. The following copyright notice applies:

The Apache Software License, Version 1.1

Copyright (c) 2000 The Apache Software Foundation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment:

"This product includes software developed by the Apache Software Foundation (http://www.apache.org/)."

Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "Apache" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org.

5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL

DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

CHAPTER 34 Index