Media Flow Controller™

Administrator's Guide

Release
12.3.2

Published: 2013-07-16

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Media Flow Media Flow Controller Administrator's Guide
Release 12.3.2, Revision 1

Revision History
3 June 2013 —Media Flow Controller Administrator's Guide

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. The Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at http://www.juniper.net/support/eula.html. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

About This Guide

This preface provides the following guidelines for using the *Juniper Networks Media Flow Controller Administrator's Guide*:

Documentation and Release Notes

For a list of related Media Flow Controller documentation see Technical Documentation page for Media Flow.

If information in the Release Notes differs from the information in this guide, follow the Media Flow Controller Release Notes.

To obtain the most current version of all Juniper Networks® technical documentation, see the product documentation page on the Juniper Networks website at http://www.juniper.net/techpubs/.

Juniper Networks supports a technical book program to publish books by Juniper Networks engineers and subject matter experts with book publishers around the world. These books go beyond the technical documentation to explore the nuances of network architecture, deployment, and administration using Junos OS and Juniper Networks devices. In addition, the Juniper Networks Technical Library, published in conjunction with O'Reilly Media, explores improving network security, reliability, and availability using Junos OS configuration techniques. All the books are for sale at technical bookstores and book outlets around the world. The current list can be viewed at http://www.juniper.net/books.

Objectives

This guide describes how to use the Media Flow Controller command-line interface (CLI) and Web interface, to configure and administer Media Flow Controller media delivery and caching.

NOTE: For additional information about the Junos OS—either corrections to, or information that might have been omitted from this guide—see the software Release Notes for your version at the Technical Documentation page for Media Flow.

Audience

This guide is designed for network system administrators who are configuring and monitoring a Juniper Networks Media Flow Controller media delivery and caching appliance.

To use this guide you need a broad understanding of networks in general, the Internet in particular, networking principles, and networking configuration. You must also be familiar with authentication scheme configurations, query parameter configurations, and media delivery protocols, such as HTTP, RTSP, RTMP, and so forth.

Supported Platforms

For the features described in this guide, Juniper Networks Media Flow Controller currently supports installation on the Juniper Networks VXA Series and generic x86 servers. For details, see the Media Flow Controller With VXA Series and Media Flow Controller datasheet.

Documentation Conventions

Table 1: Text and Syntax Conventions Used in This Book

Convention	Description	Example
Plain Text	Ordinary text.	The origin server organizes media content hierarchically.
Bold Text	Commands in running text, and screen elements such as page titles, and option labels.	Use the **interface** command to configure IP addresses. In the Management Console, use the **Setup > Date and time** page.
Italic Text	Book titles, emphasis, and variables. For variables the text must be replaced by whatever it represents. In the second example to the right, the user would replace *file_name* with the name of the specific file.	See the *Juniper Networks Media Flow Controller Administrator's Guide and CLI Command Reference* show file *file_name*
Fixed-width Text	Command keywords. Text displayed online at a command line.	interface *interface_name* Please enter your IP address
Fixed-width Bold Text	Command text that you type.	**interface eth0 ip address *IP address***

Table 1: Text and Syntax Conventions Used in This Book *(continued)*

Convention	Description	Example
[ ] (square brackets)	Optional commands. Anything not enclosed in brackets must be specified.	web proxy host *IP_address* [port *TCP_port*]
( ) (parentheses)	Represent a set of mutually exclusive options, where one option is required.	web proxy auth authtype (none | basic)
| (pipe symbol)	Separates mutually exclusive options. You can enter one of the options separated by the vertical bar, but you cannot enter multiple options in a single use of the command. A vertical bar can be used to separate optional or required options.	analytics last-evict-time diff (1 | seconds)
... (ellipsis)	An ellipsis (...) indicates that the previous option can be repeated multiple times with different values. It can be used inside or outside of brackets.	clock timezone *zone* [*zone*] ...

Documentation Feedback

We encourage you to provide feedback, comments, and suggestions so that we can improve the documentation. You can send your comments to **techpubs-comments@juniper.net**, or fill out the documentation feedback form at https://www.juniper.net/cgi-bin/docbugreport/. If you are using e-mail, be sure to include the following information with your comments:

- Document or topic name
- URL or page number
- Software release version (if applicable)

Requesting Technical Support

Technical product support is available through the Juniper Networks Technical Assistance Center (JTAC). If you are a customer with an active J-Care or JNASC support contract, or are covered under warranty, and need post-sales technical support, you can access our tools and resources online or open a case with JTAC.

- **JTAC Policies**—For a complete understanding of our JTAC procedures and policies, review the *JTAC User Guide* located at http://www.juniper.net/customers/support/downloads/710059.pdf
- **Product Warranties**—For product warranty information, visit http://www.juniper.net/support/warranty/
- **JTAC Hours of Operation**—The JTAC centers have resources available 24 hours a day, 7 days a week, 365 days a year.

Self-Help Online Tools and Resources

For quick and easy problem resolution, Juniper Networks has designed an online self-service portal called the Customer Support Center (CSC) that provides you with the following features:

- Find CSC offerings:
  http://www.juniper.net/customers/support/
- Search for known bugs:
  http://www2.juniper.net/kb/
- Find product documentation:
  http://www.juniper.net/techpubs/
- Find solutions and answer questions using our Knowledge Base:
  http://kb.juniper.net/
- Download the latest versions of software and review release notes:
  http://www.juniper.net/customers/csc/software/
- Search technical bulletins for relevant hardware and software notifications:
  https://www.juniper.net/alerts/
- Join and participate in the Juniper Networks Community Forum:
  http://www.juniper.net/company/communities/
- Open a case online in the CSC Case Manager:
  http://www.juniper.net/cm/

To verify service entitlement by product serial number, use our Serial Number Entitlement (SNE) Tool located at
https://tools.juniper.net/SerialNumberEntitlementSearch/

Opening a Case with JTAC

You can open a case with JTAC on the Web or by telephone.

- Use the Case Manager tool in the CSC at
  http://www.juniper.net/cm/
- Call 1-888-314-JTAC
  (1-888-314-5822—toll free in the USA, Canada, and Mexico)

For international or direct-dial options in countries without toll-free numbers, visit
http://www.juniper.net/support/requesting-support.html

PART 1

Configuring Media Flow Controller

CHAPTER 1

Media Flow Controller Overview

About Media Flow Controller

Juniper Networks Media Flow Controller is a purpose-built appliance that combines media intelligence, storage organization, multi-tier caching, and network optimization to scale media delivery throughput and enhance end-user experience. Media Flow Controller can be deployed by content providers, content delivery networks (CDNs), and network service providers (NSPs). Media Flow Controller can be used for origin acceleration, as well as edge or mid-tier caching, and is designed to:

- Reduce media delivery costs by providing high performance density per server.
- Scale the performance of network storages and origin servers.
- Provide a TV-like viewing experience for online video watchers.
- Save on transit bandwidth costs, accelerate Web downloads, and improve response time for requests.
- Consolidate servers by delivering mixed-media content using multiple protocols from the same server.

For more detailed information, see the Media Flow Controller product information at http://www.juniper.net/us/en/products-services/content-media/media-flow-controller/#lit.

Related Documentation

- Content Ingest Manager on page 18
- Media Flow Controller Environment on page 4
- Media Flow Controller Minimum System Requirements on page 4

Media Flow Controller Environment

Media Flow Controller software can be deployed in any network that uses the TCP/IP protocol. Media Flow Controller allows you to manage and deploy network bandwidth efficiently, thereby ensuring the highest quality experience for your end users. Media Flow Controller can be easily integrated with existing media servers, as well as management systems, for incremental and non-disruptive deployment.

- The Media Flow Controller open architecture enables it to integrate easily into existing network and storage infrastructures without requiring disruptive changes. Media Flow Controller supports industry-standard storage interfaces and devices.
- Media Flow Controller supports industry-standard video players, including Flash, QuickTime, SilverLight, and Windows Media Player.
- Media Flow Controller runs on industry-standard x86 64-bit server platforms and Juniper Networks VXA Series appliances.

  For more information, see the Media Flow Controller specifications at http://www.juniper.net/us/en/products-services/content-media/media-flow-controller/#specs.

Related Documentation

- About Media Flow Controller on page 3
- Understanding Media Flow Controller on page 5
- Media Flow Controller Environment on page 4

Media Flow Controller Minimum System Requirements

NOTE: **This section provides a high-level overview of Media Flow Controller minimum system requirements. For the most up-to-date and complete information, see the** Media Flow Controller With VXA Series and Media Flow Controller datasheet.

Media Flow Controller can run on Juniper Networks VXA Series appliances or standard x86 64-bit servers. Table 2 on page 5 shows the configuration that is either required or recommended for optimal performance when running the Media Flow Controller software on generic (non-Juniper Networks) x86 servers.

Table 2: Media Flow Controller Recommended Hardware Configuration

Item	Reverse Proxy	Transparent Proxy
Processor	64-bit x86, Dual Quad Core or more, minimum of 2.4 GHz speed	64-bit x86, Dual Quad Core or more, minimum of 2.4 GHz speed
RAM	8 GB or more	36 GB or more
Direct Attached Storage	Up to 16 direct attached storage (DAS) drives such as SATA, SAS, or SSD, configured as JBODs (just a bunch of disks)	
Solid State Drives	Intel Extreme, or Intel Mainstream, for higher performance	
Disk Controllers (RAID should be disabled.)	Dell SAS 6/ir, HP SC44Ge LSI SAS 3442E-R/3081E-R LSI 1068/1064/1078 LSI SAS2008 LSI Logic or Symbios Logic SAS1068E PCI-Express Fusion-MPT SAS HP Smart Array G6 controllers (rev01) 3Ware 9690SA	
Network Controllers	Intel 1 GbE (82574, 82580, 82571 (will be EOLed), 82576, 82575eb), Intel 10 GbE 82599 Chelsio 10 GbE	
Network Interfaces	One 100 Mbps or 1 Gbps management interface Up to 10 x 1 Gbps or 2 x 10 Gbps media delivery interfaces	

Related Documentation

- *Pre-installation Planning*
- *Example: Machine Setup*

Understanding Media Flow Controller

Media Flow Controller operates in three different proxy modes: reverse proxy, transparent proxy, and mid-tier proxy. For more information, see "Media Flow Controller Deployment Guidelines" on page 447.

Media Flow Controller consolidates all streaming protocols (HTTP, RTSP, and RTMP) into a single server, thereby reducing the number of servers required to deliver video over multiple protocols.

Media Flow Controller obtains content from origin servers or origin storages once, and serves it to several users simultaneously.

When a user request is received, Media Flow Controller checks for the presence of the object in its cache file system. If no copy exists in any cache (also known as "cache miss"), Media Flow Controller sends a request to the target origin server, fetches the content, and serves it to the user. Then Media Flow Controller decides whether that content should be cached. Media Flow Controller decides whether to cache content based on its intelligent Analytical Engine, request and response headers, and customer-configured policies.

Media Flow Controller keeps track of the popularity of objects in the cache. The popularity of the object is calculated based on the frequency of hits (the number of repeat requests to an object in a given time period). When objects become " hot" (popular), Media Flow Controller promotes them to a cache tier that supports faster delivery. Promotion in Media Flow Controller begins from the lowest tier; for example, SATA to SAS, SSD, and RAM. This allows Media Flow Controller to scale throughput and meet increased demand.

Figure 1: Juniper Networks Media Flow Controller Operations (Reverse Proxy Deployment)

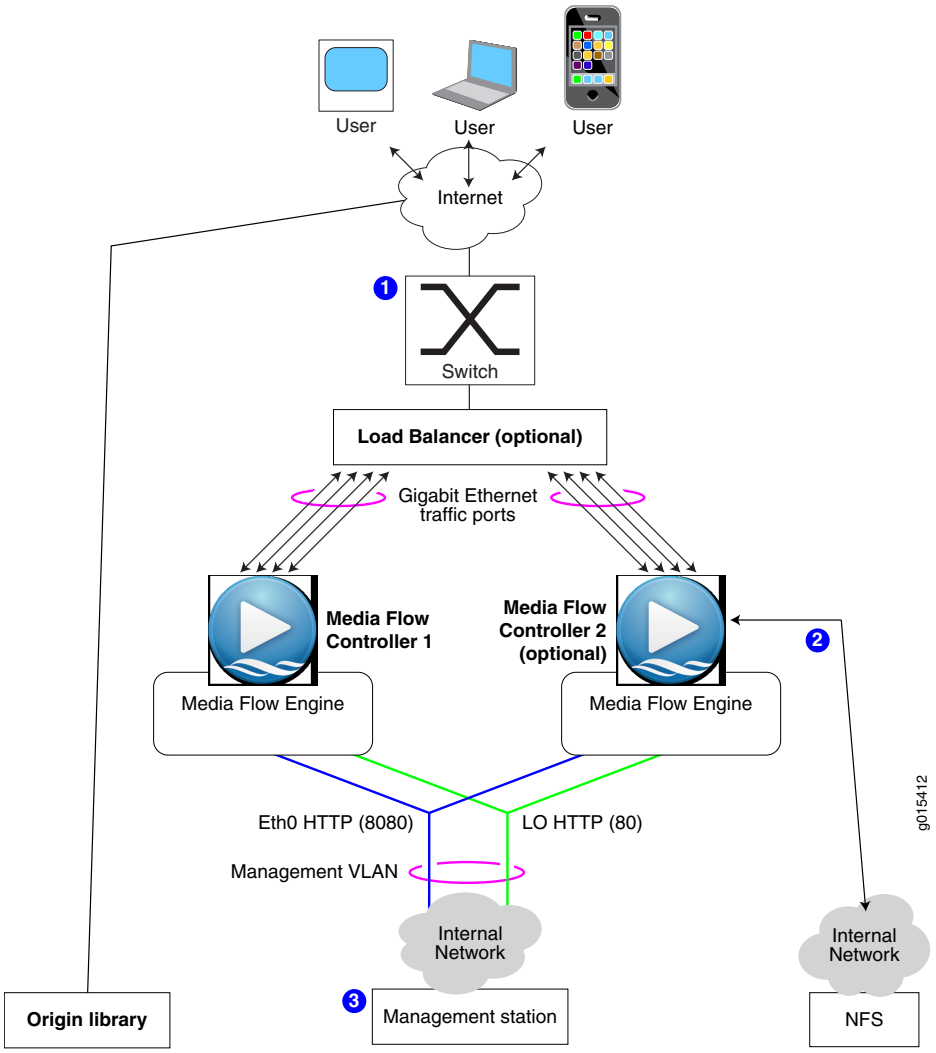


Figure 1 on page 6 illustrates the media delivery optimization operation.

1. Requests come in from the Internet using HTTP to (typically) a switch or load balancer that redirects the request to Media Flow Controller. Media Flow Controller does certain basic checks, such as URL validation, and parses the URL query string and header fields to identify the content and any associated policies. It alsodoes a resource check to verify that the content can be delivered in an acceptable manner for that session.
2. When a cache miss occurs (content is not cached), Media Flow Controller gets the content from the origin, serves it, and caches a copy. For objects larger than 32 KB, serving begins when 32 KB are received; for objects smaller than 32 KB, serving begins at 4 KB or when the object is received. Subsequent requests for the same content are served directly from the cache.
3. Management interfaces monitor activity and allow configuration changes.

Related Documentation

- About Media Flow Controller on page 3
- Media Flow Controller Environment on page 4

Media Flow Controller Management Interfaces

Media Flow Controller can be configured and managed through a variety of interfaces.

- Command-Line Interface (CLI) on page 7
- Web Interface (Management Console) on page 7
- SNMP Agent Support on page 8
- E-mail and E-mail2SMS Alerts on page 8
- Logs on page 8

Command-Line Interface (CLI)

You can log in to the Media Flow Controller server using the command-line interface (CLI) by invoking an SSH session. The CLI allows all aspects of system configuration and management. For more information about configuring Media Flow Controller using the CLI, see the "About the Media Flow Controller CLI" on page 21. For more information about the Media Flow Controller CLI, see the *Media Flow Controller CLI Command Reference.*

Web Interface (Management Console)

You can log in to the Media Flow Controller Web interface, also referred to as the Management Console, from a Web browser using HTTP and port 8080 (for example, **http://*Media Flow Controller-Hostname*:8080/**). HTTPS can also be used if SSH certificates are set up properly. The Web interface makes it easy to configure and manage the system from any remote location. The Web interface provides a powerful dashboard that displays real-time performance data such as the number of concurrent connections, cache-hit ration, bandwidth served, and CPU/memory/disk utilization. For more information about using the Media Flow Controller Web interface, see "About the Media Flow Controller Web Interface" on page 281.

SNMP Agent Support

Media Flow Controller includes an SNMP agent for monitoring various system statistics and parameters. The Media Flow Controller SNMP agent supports only monitoring and does not support any network configuration features.

For more details, see "SNMP Support Overview" on page 401.

E-mail and E-mail2SMS Alerts

Media Flow Controller notifies you using e-mail during events such as high CPU or memory utilization, interface up or down, and threshold crossing on statistics or counters. Media Flow Controller uses the SMTP protocol to send e-mails to the administrators. You can use the E-mail 2SMS facility provided by mobile network operators to configure Media Flow Controller to send SMS notifications.

```
From: "System Administrator" <do-not-reply@mfc.example.com>
Date: December 28, 2009 9:40:47 AM PST
To: admin@example.com
Subject: System event on mfc.example.com: Process exit: ftpd
Hostname: mfc.example.com
Date: 2009/12/28 17:40:47
Description: Unexpected exit of process ftpd.
Uptime: 1h 15m 34.860s
Version: mfc-1.2.0
```

For more information, see "Configuring Media Flow Controller Fault Notifications (CLI)" on page 372 and "Configuring Faults and Logging (Web Interface)" on page 309.

Logs

Media Flow Controller generates *system logs* for system events such as interface up/down, user login, configuration logging, software process crashes, and so forth.

Media Flow Controller generates error logs for debugging and troubleshooting internal system errors.

Media Flow Controller generates W3C/NCSA-compliant access logs to track all media requests. Media Flow Controller also allows administrators to customize the format of the access log entries. Media Flow Controller can be integrated with third-party reporting tools (such as Sawmill) to generate reports for analyzing usage patterns and for capacity planning. For more information about Media Flow Controller logs, see "Media Flow Controller Logging Overview" on page 332 and "Configuring Faults and Logging (Web Interface)" on page 309.

Related Documentation

- About Media Flow Controller on page 3
- SNMP Support Overview on page 401
- Understanding Media Flow Controller on page 5

Media Flow Controller Delivery Methods

Understanding Media Flow Controller Delivery

Media Flow Controller can deliver content simultaneously to a large audience across multiple screens (such as PCs, TVs, and mobile devices) by supporting a wide range of delivery protocols and media formats.

Media Flow Controller consolidates multiple delivery protocols such as HTTP, HTTPS, RTSP, and RTMP; see Table 3 on page 9. Media Flow Controller can be used for delivering rich media content to users, including streaming of both on-demand and live videos, . See Figure 2 on page 10.

Media Flow Controller:

- Efficiently caches objects of all sizes, ranging from small objects (thumbnails) to the largest objects (videos and software downloads)
- Supports delivery using HTTP, HTTPS, RTSP, and RTMP
- Supports various formats required for delivery to different screens
- Supports multi-tenancy to host content belonging to multiple customers

Media Flow Controller supports the entire spectrum of adaptive streaming methods for on-demand and live streaming such as Apple iPhone Streaming, Microsoft Smooth Streaming, Move Adaptive Streaming, and Adobe Dynamic HTTP and RTMP streaming.

Media Flow Controller supports HTTP Progressive Download (PDL).

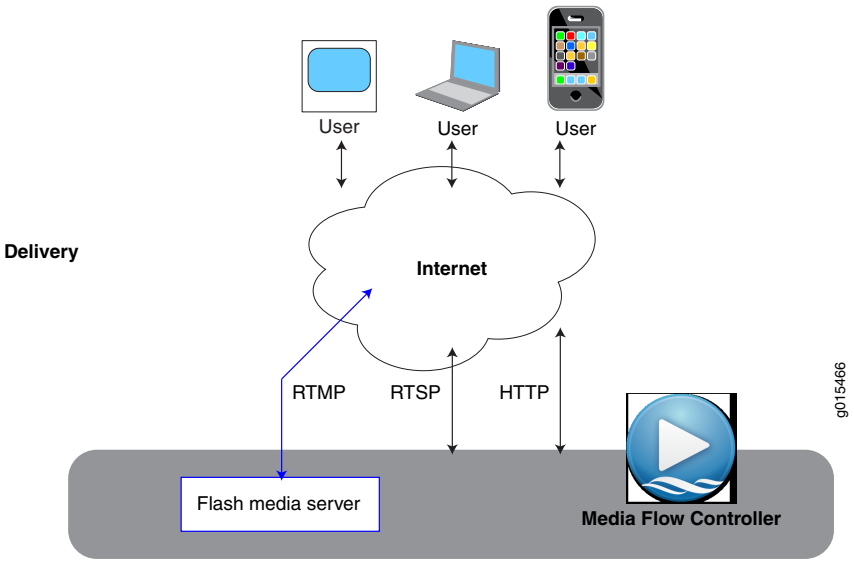
Table 3: Protocol Support Matrix

Content Ingest or Publishing Protocol	Delivery Type	Delivery Protocols
HTTP Fetch	On-demand and live	HTTP, RTP/RTSP*, RTMP*
NFS Fetch (file system containing assets mounted on a Media Flow Controller)	On-demand or live	HTTP, RTP/RTSP*, RTMP*
FTP (push)	On-demand	HTTP, RTP/RTSP, RTMP
RTP and RTSP Ingest	On-demand and live	RTP and RTSP
RTMP Ingest	On-demand and live	RTMP

Table 3: Protocol Support Matrix *(continued)*

Content Ingest or Publishing Protocol	Delivery Type	Delivery Protocols
* Only on-demand media delivery is supported for this combination.		

Figure 2: Media Flow Controller Delivery Options



Media Delivery Using HTTP

Media Flow Controller supports delivering rich media content to users with the HTTP protocol. The HTTP protocol can be used for delivering static Web objects such as HTML, images, and thumbnails; and for streaming videos. Media Flow Controller also supports caching and delivery of HTML5 videos when progressively downloaded using HTTP. For HTTP, the supported methods are CONNECT, DELETE, GET, HEAD, OPTIONS, POST, PUT, and TRACE.

Media Flow Controller supports HTTP pipelining requests from clients. The pipelined requests are served from the cache, if the content is available in the cache. HTTP pipelining support is enabled by default. There is no configuration required for this feature.

Media Delivery Using RTSP

Media Flow Controller improves the performance of media delivery using the Real-Time Streaming Protocol (RTSP), by using its caching and network scaling infrastructure. Media Flow Controller supports the trick-play/DVR-like functions based on the RTSP protocol Pause and Seek operations.

NOTE: Trick-play capability is supported for H.264/AAC encoding and MP4 format, but is not currently supported for WMP/WMV using VC1 encoding or proprietary formats.

Media Flow Controller uses a suite of protocols to deliver videos using RTSP:

- RTSP for establishing and controlling media sessions with the player.
- RTP to deliver the actual audio and video streams.
- RTCP to provide control information for an RTP flow and statistics for a media connection.

Media Flow Controller provides options for using both UDP or TCP protocols for transmitting RTP streams. You can use interleaved RTSP to effectively send both control information and data streams on the same transport connection. This reduces the number of ports used for transmitting streams to a player, thereby reducing the resource utilization on firewalls and switches.

The native RTSP implementation in Media Flow Controller Release 12.1 interoperates with Quicktime players and VLC players. The supported methods for RTSP are DESCRIBE, OPTION, PAUSE, PLAY, SETUP, and TEARDOWN; all are required except OPTION and PAUSE, which are recommended.

NOTE: **Unlike HTTP, there are multiple implementations of RTSP. The three most popular implementations are from Apple, Real Networks, and Microsoft. Each of these claim to be RFC compliant and yet are different due to their proprietary features such that they do not interoperate with each other.**

Media Delivery Using HTTPS

Media Flow Controller supports media delivery using Hypertext Transfer Protocol over SSL (HTTPS) to users. It fetches media from origin servers using HTTP/NFS and delivers using HTTPS. HTTPS is the standard encrypted communication mechanism on the Web.

SSL allows clients to connect to Media Flow Controller securely. SSL uses public key encryption to authenticate the server to the client. SSL support is a separately licensed feature. You must install an SSL license before you can configure SSL parameters.

Use HTTPS support in Media Flow Controller for media delivery in Reverse Proxy deployments. HTTPS support in Media Flow Controller provides the following benefits:

- Consolidates the delivery of multiple protocols, including HTTP, HTTPS, RTSP, and RTMP from the same server, thereby reducing the number of servers used for media delivery.
- Eliminates the SSL processing overhead from origin servers.

Adaptive Bit Rate Streaming

Adaptive Bit Rate Streaming technology requires content to be encoded at multiple bit rates and then delivered to clients as a series of small chunks or fragments. This allows the client player to dynamically switch between fragments of different bit rates depending on the network bandwidth, CPU state, and so forth, allowing viewers to have the best possible viewing experience.

Media Flow Controller can improve the media delivery throughput of other proprietary HTTP-based streaming technologies such as Adobe Flash Streaming, Move Streaming, Apple HTTP Streaming, and Microsoft Smooth Streaming. Media Flow Controller can be used to deliver both on-demand and live video streams to Move player, Apple iPhone/iPod Touch, Adobe Flash, and Microsoft Silverlight players.

Smooth Streaming is an HTTP-based adaptive-streaming technology implemented by Microsoft. The media format defined by Microsoft for smooth streaming supports both storage and on-the-wire delivery, and is based on the ISO/IEC 14496-12 ISO Base Media File Format specification. Similarly, Apple has defined an HTTP Live Streaming (HLS) IETF RFC based on MPEG-2 TS stream segments, along with an extension to their M3U8 playlist format, to support adaptive stream delivery to iPhone Operating System (iOS) devices. Adobe also recently released an HTTP Dynamic Streaming (HDS) variant for their Flash Player, which is also derived from the ISO/IEC 14496-12 Base Media File Format.

Media Flow Publisher provides the capability to publish on-demand and live content to the various adaptive HTTP streaming technologies. Media Flow Controller supports delivering pre-segmented videos over HTTP using adaptive streaming techniques.

Related Documentation

- About Media Flow Controller on page 3
- Media Flow Controller Environment on page 4
- Understanding Media Flow Controller on page 5

Caching and Origin Clustering

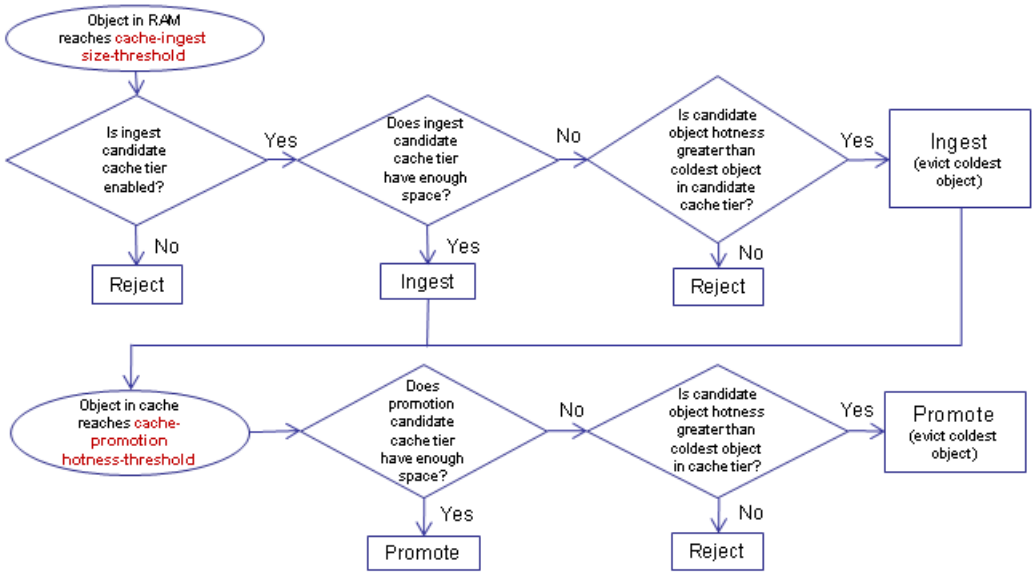
- Media Flow Controller Hierarchical Caching on page 12
- Factors Influencing Whether an Object Will Be Cached on page 13
- Connection Pooling on page 14
- Origin Clustering and Origin Escalation on page 14

Media Flow Controller Hierarchical Caching

When Media Flow Controller fetches data from origin upon cache miss, it caches the data in its local disks. Media Flow Controller has its own optimized storage sub-system in which data is placed intelligently so it can be read back for very fast delivery. Media Flow Controller organizes data in a hierarchical fashion using a cache tier manager that dynamically calculates the "hotness" of the data and places it in the correct cache tier. RAM is the highest tier, followed by SSD, SAS, and SATA, in that order.

Disk speeds are calculated and assigned to a tier in the cache hierarchy as part of the initialization of Media Flow Controller. When data is accessed from origin, it is stored in the lowest cache tier, and promoted to higher cache tiers as the hotness of the data increases. See Figure 3 on page 13.

Figure 3: Media Flow Controller Cache Ingest and Promotion Process



Factors Influencing Whether an Object Will Be Cached

The following are common conditions under which a request or response is not cached by Media Flow Controller:

- Origin server does not return either a 200 or 206 response code.
- Request URL has a query string and “disable cache on query string” is enabled under the **namespace** configuration.
- Origin server returns a 206 response and the content is marked as “chunked encoding.”
- Origin server does not return content-length, or “chunked encoding” is not present in the response.
- Origin server returns a content-length with a value of 0 (zero).
- Request URL contains hexadecimal characters.
- Calculation of the object version fails. To determine whether a response object is the same as a cached object, Media Flow Controller adds version control to each response object. Every object version is calculated based on the domain, URI, ETag, Date, and Last-Modified-Date fields from the HTTP transaction. If object versions are different, the object has been modified and is not cached.

NOTE: Starting with Release 12.2, Media Flow Controller caches objects with Cache-Control: no-cache. The object will be revalidated with the origin before being served to the user.*

- “Cache-control: no-cache” header is present in the request or response.*
- Content-Range request and response do not match the offset or length of the object.

- Request or response contains the following headers:**
  - Cache-Control: Private
  - Cache-Control: no-cache*
  - Cache-Control: No-Store
  - Cache-Control: Max-Age = 0*
  - Pragma: No-Cache
- Object is expired (for example, an object's expiry time is less than the Media Flow Controller current system time, or an object's age has exceeded the configured age time).**
- Response contains cookies (Set-Cookie/Set-Cookie2) and " disable cookie caching" is enabled in the **namespace** configuration.**
- Analytics Manager decides that the object is not worthy of caching because the size is below that configured by the **namespace** command **delivery protocol http origin-fetch content-store media object-size** ***size***.

** Can be overridden by using the **namespace** configuration to forcefully cache objects that are marked as " no-cache."

When Media Flow Controller decides that a request or response is non-cacheable, the response is tunneled through. The response is not stored in the cache file system.

The response is cached when any of the preceding conditions are not encountered.

Related Documentation

- Caching Client Requests Using Tunnel Override on page 83

Connection Pooling

Media Flow Controller fetches content from an origin server if content is not present in the cache. While fetching content from origin, Media Flow Controller can minimize the load on origin servers by pooling TCP connections. Media Flow Controller establishes a fixed number of connections with the origin server and reuses them for new requests. This minimizes the connection setup overhead and also reduces the load on origin servers. Connection pooling is beneficial when Media Flow Controller is deployed as an origin accelerator.

Origin Clustering and Origin Escalation

Media Flow Controller supports **origin-server** node map configuration for origin escalation: if the target origin server fails, another configured origin server is automatically chosen. These configurations are achieved through the creation of a **server-map** (**format-type cluster-map** and **format-type origin-escalation-map**) that is then associated with a namespace.

Related Documentation

- *logical-interface (DDoS Flow Detection)*
- Origin Server Load Distribution and Failover on page 225

- About Media Flow Controller on page 3

Media Flow Controller Multi-Tenancy Management

Multi-tenancy allows the hosting of multiple websites or domains on a single Media Flow Controller using the **namespace** and **resource-pool** entities. The multi-tenancy feature in Media Flow Controller can be used by:

- Content Delivery Networks (CDNs) to host websites belonging to multiple customers on a single Media Flow Controller.
- Content Publishers to host multiple websites or domains on a single server.

A tenant can be a customer or a Web portal, and is represented in Media Flow Controller using one or more namespaces. Resources can be assigned to each tenant and resource usage can be monitored. Media Flow Controller provides an entity called resource-pool; you use the **resource-pool** command to configure resources that are relevant for a tenant such as bandwidth, and maximum allowed sessions. When you create a resource pool and bind it to a namespace, it sets the configured limits on the various resources for the tenant. The number of viewers, or end users, allowed on a system rely on the resource-pool allocations.

The following are some use-cases of achieving multi-tenancy with Media Flow Controller:

- A CDN service provider can allocate, monitor, and limit resources for a large customer running a video portal, a mid-size customer running a news portal, and a small customer's corporate website, all from the same Media Flow Controller. You achieve this by provisioning a namespace, creating and setting limits on a resource-pool, and binding the namespace to the resource pool.
- A Content Publisher can limit the number of users watching video from personal computers versus mobile devices versus television. You achieve this by mapping users to a namespace based on the "User-Agent" request header, creating and setting concurrent session limits on a resource-pool, and binding the namespace to the resource pool.
- A Content Publisher can limit the amount of bandwidth consumed by local users versus international users. You achieve this by mapping users to a namespace based on the query parameter containing "GEO" information, creating and setting concurrent session limits on a resource pool, and binding the namespace to the resource pool.

NOTE: **Resource pools can be applied when delivering media using the HTTP protocol.**

Related Documentation

- Understanding Media Flow Controller on page 5
- About Media Flow Controller on page 3

Media Flow Controller Namespaces

The **namespace** function allows you to classify different types of traffic based on a combination of URL and fully qualified domain name (FQDN), and apply separate delivery policies to each type of classified traffic. This gives you a way to separate your video-delivery traffic characteristics based on any given variable in the stream/request being received by Media Flow Controller from the client. You can create up to 256 namespaces in one Media Flow Controller.

At a minimum, **namespace** configuration requires a **domain** (only one per namespace), an **origin-server** (one per namespace unless using **server-map**), and a **match** criteria (to refine delivery of incoming requests). Additional parameters for **origin-fetch**, **cache** options, and so forth, are optional. You can further define control by assigning a configured **virtual-player**. The namespace is referenced using the URL in the HTTP request to Media Flow Controller.

A namespace usually represents one or more Web properties or websites. A CDN service provider can define one or more namespaces for a given CDN customer. For example, a CDN customer might have multiple domains such as www.foo.com, www.bar.com, and www.foobar.com. If the media caching and delivery policies are the same across all the three properties, a CDN service provider can just define one namespace for the CDN customer. If the media caching and delivery policies are different for the three Web properties, then separate namespaces have to be defined in the Media Flow Controller.

A content publisher can define one or more namespaces to represent a website. For example, a content publisher may serve videos and Web objects from the same website. However, the delivery protocols or policies for videos might be different from the delivery protocols or policies for the Web objects. In such a scenario, a content publisher defines separate namespaces for videos and Web object delivery.

Related Documentation

- Using Namespace for Dynamic URI Remapping on page 113
- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Differentiated Services Code Point Marking and Cache Control Overview on page 115
- Understanding Media Flow Controller on page 5

Media Flow Controller Virtual Players

The virtual player function contains a set of policies that allow you to control the delivery of media. Virtual player provides an infrastructure to perform the following functions:

- Uniquely identify objects in the cache, although the objects might be represented using different URLs
- Provide trick-play functions such as forward, rewind, seek, and pause
- Guarantee bandwidth for each user session
- Validate URLs before serving user requests

You can create any number of virtual players; they are utilized when assigned to a namespace. Namespaces that are not assigned a **virtual-player** use the values configured under **network connection** for the same functions. Virtual players let you implement custom delivery policies.

Related Documentation

- About Media Flow Controller on page 3

Media Flow Controller Policy Engine

Media Flow Controller allows fine-grained control over caching and delivery decisions through a programmable interface for policy definition called Policy Engine. Policy Engine provides maximum flexibility to define customized policies using a powerful combination of regular expressions, conditional statements, and access to HTTP request and response headers. Use the Policy Engine to perform tasks such as the following:

- Differentiate services by prioritizing customers or content types using Terms of Service (ToS) and Differentiated Services (DSCP) marking.
- Select origin servers based on content type or request headers. For example, content providers can configure Media Flow Controller to serve mobile users from one origin and personal computer users from a different origin.
- Redirect users to alternative websites or pages, when servers are busy or under maintenance.
- Prevent content thefts by selectively blocking users, based on geographical location of the user, an IP address or subnet, or HTTP referrer header.
- Flexible policy definitions using REGEX.

Related Documentation

- Understanding Media Flow Controller on page 5

Media Flow Controller Media Flow Publisher

Media Flow Publisher compliments Media Flow Controller's adaptive stream caching capabilities by providing adaptive stream segmentation, stream packaging, metadata publishing, and adaptive stream format translation. Media Flow Publisher supports industry leading adaptive stream formats from Apple, Adobe, and Microsoft on a single platform for both live video streams (for example, live TV) and on-demand file sources.

Media Flow Publisher eliminates the need to deploy specialized adaptive stream encoders and servers by converging the segmentation, packaging, and metadata publishing for Juniper Networks, Apple, Adobe, and Microsoft adaptive stream formats all on a single platform.

Related Documentation

- Understanding Media Flow Controller on page 5

Content Ingest Manager

The Content Ingest Manager allows you to keep your content fresh by periodically searching content provider repositories or parent caches and downloading fresh content to your parent and edge Media Flow Controller caches using a scheduled HTTP crawler program. The following are the benefits of Content Ingest Manager:

- Automatic download of content from origin file servers at scheduled intervals. It is not necessary for you to initiate the propagation of content across the Content Delivery Network (CDN).
- You are always served fresh content. Cached content is kept fresh by periodically refreshing it from origin file servers.
- Almost all the requests for content are served from the edge cache to improve the user experience.

Related Documentation

- Content Ingest Manager Overview on page 171
- About Media Flow Controller on page 3

CHAPTER 2

Configuring and Administering Media Flow Controller (CLI)

Before You Configure Media Flow Controller

As with any network appliance, the Media Flow Controller network parameters must be configured first. Before you begin, you need to know:

- Hostnames and IP addresses (including but not limited to, subnet mask, default gateway, DNS servers, and NTP servers) for the Media Flow Controller traffic and management ports (Eth0 is the default management port, and Eth1 is the recommended origin-fetch interface).
- Hostnames and IP addresses for external servers such as origin servers or libraries, logging, SNMP, SSH, or storage servers. In order to configure required namespaces, you must know the uri-prefix, domain name, and origin server's fully qualified domain name (FQDN) or IP address, at a minimum. This information tells Media Flow Controller where to fetch media.
- Domain Name Servers need to be configured for the Media Flow Controller.
- The users that will be able to administer or monitor Media Flow Controller, their e-mail addresses (for event e-mail notifications), and the authentication or authorization schemes you want to use; these schemes can be complicated and should be prepared by an expert.
- The query parameters that you use in URLs to pass information, if you expect to configure a virtual player (not required). Many Content Delivery Networks (CDNs) have proprietary query parameters (also known as query params) already defined.
- The types of content that you serve and their optimal delivery rates, the protocols that will be used to fetch the content, and the general bandwidths of delivery connections that you want to maintain.

Related Documentation

About the Media Flow Controller CLI

The Juniper Networks Media Flow Controller command-line interface (CLI) supports industry-standard commands for configuration and management, as well as Media Flow Controller-specific commands.

The CLI supports command-line editing: press the up arrow to repeat previous lines, and the left arrow to edit the current line. The CLI also supports command completion when you press the Tab key. Commands must terminate with a carriage return followed by newline (CRLF).

Connecting and Logging In

You can connect to the CLI with SSH or telnet (after enabled; telnet is disabled by default) using the IP address of your Media Flow Controller, or you can connect directly to the serial console. The serial console does not require the IP address of the Media Flow Controller; it requires physical access to the hardware and a serial cable connection. The Media Flow Controller responds with a login prompt. Enter **admin** as the user and leave the password blank; there is no default password. After you have connected, enter **enable** and then **configure terminal** to begin configuring Media Flow Controller.

Likewise, you can log in to the Web-based interface by entering the IP address in a browser window and using **admin** as the login name. The Management Console has a subset of the CLI commands, and is adequate for simple or first-day configurations.

Each user account has at least one privilege level that determines which commands the user can issue and what CLI modes the user can access:

- Administrator (**admin**)—Full privileges. Can enter **Enable** mode and **Config** mode.
- Monitor (**monitor**)—Can read data (not including logs) and perform actions, but cannot change any configuration. Can enter **Enable** mode from **Standard** mode, but cannot change configurations.
- Unprivileged (**unpriv**)—Can issue a small subset of commands, including debugging and show commands. Can log in to **Standard** mode only.

Using the Command Modes

When you log in to the management shell over SSH (or optionally telnet, if enabled; this is not recommended) you are in the lowest tier or **Standard** mode, where you can enter only **show**, **help**, **diagnostic**, and a few others other commands. You access **Enable** mode by issuing the **enable** command. In **Enable** mode you can view current configurations but you cannot work with configurations. You must enter **Configure** mode to make any

changes. The CLI can be in one of three modes, which determine which set of commands is available:

- **Standard** mode—When the CLI is launched, it begins in **Standard** mode. This is the most restrictive mode, offering basic system commands that query a restricted set of state information. You cannot alter the system or change settings.
- **Enable** mode—The **enable** command accesses **Enable** mode. This mode has privileged system commands to view all state information, and take certain kinds of actions such as rebooting the system; however, you cannot make any configuration changes in this mode. Enable mode commands are a superset of those in Standard mode. Enter **disable** to exit **Enable** mode.
- **Configure** mode—The **configure terminal** command moves you to **Configure** mode. This has a full unrestricted set of commands with which you can view anything, take any action, or change any configuration. The Configure mode commands are a superset of those in **Enable** mode. Enter **exit** to leave **Configure** mode.
- **Prefix** mode—Some commands have a **prefix** mode; that is, when you enter a keyword, you enter a mode for that configuration. For example:

  MFC (config) # accesslog
  MFC (config accesslog) #

When in **prefix** mode, you can only make configurations for that command set; entering **?** (question mark) shows only the options for those configurations. To leave the **prefix** mode, enter **exit**.

Example:

```
MFC >          enable
MFC #          configure terminal
MFC (config) #      namespace test
MFC (config namespace test) # exit
MFC (config) #       exit
MFC #          disable
MFC >
```

Prompt and Response Conventions

The prompt always begins with the hostname of the system. What follows depends on what command mode you are in. For example, say the hostname is "MFC." The prompts for each mode would be:

```
MFC >      Standard mode
MFC #      Enable mode
MFC (config) #   Configure mode
```

Successful commands do not return a response. When you press Enter, you see the command prompt. If an error is encountered when executing a command, the response begins with % (percent sign), followed by text describing the error.

NOTE: All CLI commands allow completion with the Tab. For example, typing **en** and then pressing the Tab completes the **en** command to **enable**. Completion (pressing the Tab) also shows all commands after the typed letters; for example, typing **e** (in Standard mode) and then pressing the Tab key shows **enable** and **exit** as the available commands starting with **e**.

CLI Options

Four groups of commands relate to the CLI itself:

- **cli session** commands change a setting only for the current CLI session. They do not affect any other sessions, and can be performed by any user at any time.
- **cli default** commands change the defaults for the specified setting for all future CLI sessions of all users. They also change the setting for the current session from which they were executed, but not for any other currently active sessions. Since they change configuration, the user must be in configuration mode to run them, and they can only be run by a user with **admin** privileges.
- Other **cli** commands that take one-time actions, instead of changing a setting. These commands do not fall under the session or default umbrellas; for example, **cli clear-history**.
- **terminal** commands are clones of a subset of the **cli session** commands, and are only present for convenience.

See **cli** in the *Media Flow Controller CLI Command Reference* for CLI command details.

NOTE: Some settings, such as the terminal length and width, are session-specific, and have no corresponding commands to set defaults. Also, some commands are only available in default form.

Related Documentation

- Before You Configure Media Flow Controller on page 20
- Understanding Media Flow Controller on page 5
- About Media Flow Controller on page 3

Logging In to Media Flow Controller for the First Time

Before you log in to Media Flow Controller for the first time, see "Before You Configure Media Flow Controller" on page 20. Also make sure you have the IP address assigned to the management interface.

To log in to the system command-line interface (CLI) for the first time:

1. Open an SSH session and enter the Media Flow Controller management IP address or hostname, or open a serial console session with the console server IP address and port.
2. Log in with the default credentials (there is no default password).
   User: **admin**

To log in to the Management Console (Web UI):

1. Using a browser, navigate to the configured Media Flow Controller IP address on the management port (**:8080**). Example:

   http://192.168.1.100:8080.
2. Log in with the default credentials (there is no default password).
   User: **admin**

   The Management Console has a subset of the CLI commands, and is good for simple or first-day configurations.

Related Documentation

- Before You Configure Media Flow Controller on page 20
- Understanding Media Flow Controller on page 5
- About the Media Flow Controller CLI on page 21

Using SSH in Automated Scripts (CLI)

SSH clients such as, **ssh**, **sftp**, and **scp** require authentication for each session. By default, these clients use password authentication. To automate authentication for trusted users and hosts, including Media Flow Controller hosts, you must exchange host keys between each host pair.

For example, **accesslog copy** can use **scp** or **sftp** to automatically upload log files. However, to enable Media Flow Controller to automatically upload log files without password authentication, the SSH utilities must be configured to use host key-based authentication. You can identify a machine in the network and its trusted user who can execute commands without entering a password.

To allow SSH command execution for a user on a trusted host:

1. Identify a trusted machine in the network.
2. Generate an ssh key for the trusted user on the Linux server using ssh-keygen.

```
aspag@weborigin:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/aspag/.ssh/id_rsa):
Created directory '/home/aspag/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/aspag/.ssh/id_rsa.
Your public key has been saved in /home/aspag/.ssh/id_rsa.pub.
The key fingerprint is:
4c:d3:b3:a3:91:7a:ac:2b:74:78:19:48:75:78:fa:26 aspag@weborigin
The key's randomart image is:
```

```
+--[ RSA 2048]----+
|     ....        |
|    . ....       |
|   . . oo o      |
|    . oo o o     |
|     . +S o      |
|    o Eooo .     |
|   . o.o+        |
|    .  o         |
|     .o.         |
+-----------------+
```

3. Verify public key for trusted user on Linux server.

```
aspag@weborigin:~$ cat /home/aspag/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAsPWL1dU6jRkL2j9KhllgbtY1
oZWr8bAbL9x6jTtgMMCAOiOmddk3FFYBQulFFYm7XsIjtopBkd25SOsi1T7k455EHb
2x9ciPeO8s8NCeTinOzTI2ZDjHgfM21C38Zu/zzxdBl29AOg7pCdl54BljZv9tcLOG6L
2r4g83OVUNFDmG4p2qNM3uh33AJAH+LlykBJeCsc0qBJAAnG6eb51jkXb0bKlXtcb2
T1BSXQEDKHFiLjBHdrkHUE12vjXOvIdmv+bbTMSMERRT9SuR+C+pXRHQ5jYTxDoP3YVe
ZUDxOaorxvzpT9R4+2mpXRsSGmhU4cAvqY/06yUdxOXa/v4lVw== aspag@weborigin
```

NOTE: The key begins with ssh-rsa and ends with ==trusted user@known host. The entire content starting with ssh-rsa and ending with aspag@weborigin must be included within the quotes when the key is added to the software.

4. Create an account for the trusted user on the software.

```
LBS-NGEMFC2-1001 (config) # username aspag
LBS-NGEMFC2-1001 (config) # username aspag password 0 321aspag
LBS-NGEMFC2-1001 (config) # username aspag capability admin
LBS-NGEMFC2-1001 (config) # show usernames
USERNAME    FULL NAME               CAPABILITY  ACCOUNT STATUS
admin       System Administrator    admin       Password set
anthonytest_ftpuser                 ftpuser     Local password login disabled
aspag                               admin       Password set
aspagnolo                           admin       Password set
cmcrendv    CMC Rendezvous User     cmcrendv    Local password login disabled
mfc_probe_ftpuser                   ftpuser     Local password login disabled
monitor     System Monitor          monitor     No password required for login
ngemfc2_test0_ftpuser               ftpuser     Local password login disabled
ngemfc2_test1_ftpuser               ftpuser     Local password login disabled
ngemfc2_test2_ftpuser               ftpuser     Local password login disabled
ngemfc2_test3_ftpuser               ftpuser     Local password login disabled
```

5. Configure the software to use a public key for the trusted user from a known host.

```
LBS-NGEMFC2-1001 (config) # ssh client user aspag authorized-key sshv2
"ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAsPWL1dU6jRkL2j9KhllgbtY1
oZWr8bAbL9x6jjTtgMMCAOiOmddk3FFYBQulFFYm7XsIjtopBkd25SOsi1T7k455EHb
2x9ciPeO8s8NCeTinOzTI2ZDjHgfM21C38Zu/zzxdBl29AOg7pCdl54BljZv9tcLOG6L
2r4g83OVUNFDmG4p2qNM3uhh33AJAH+LlykBJeCsc0qBJAAnG6eb51jkXb0bKlXtcb2
T1BSXQEDKHFiLjBHdrkHUE12vjXOvIdmv+bbTMSMERRT9SuR+C+pXRHQ5jYTxDoP3YVe
ZUDxOaorxvzpT9R4+2mpXRsSGmhU4cAvqqY/06yUdxOXa/v4lVw== aspag@weborigin"
```

6. Verify configuration of the key on the Media Flow Controller.

```
LBS-NGEMFC2-1001 (config) # show ssh client
```

```
SSH client Strict Hostkey Checking: ask

No SSH global known hosts configured.

No SSH user identities configured.

SSH authorized keys:
  User aspag:
    Key 1: ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAsPWL1dU6jRkL2j9KhllgbtY1oZWr8bAbL
9x6jTtgMMCAOi0mddk3FFYBQulFFYm7XsIjtopBkd25SOsi1T7k455EHb2x9ciPe
O8s8NCeTinOzTI2ZDjHgfM21C38Zu/zzxdBl29AOg7pCdl54BljZv9tcLOG6L2r4g
830VUNFDmG4p2qNM3uh33AJAH+LlykBJeCscOqBJAAnG6eb51jkXbObKlXtcb2T1BSXQEDK
HFiLjBHdrkHUE12vjXOvIdmv+bbTMSMERRT9SuR+C+pXRHQ5jYTxDoP3YVeZUDxO
aorxvzpT9R4+2mpXRsSGmhU4cAvqY/O6yUdxOXa/v4lVw== aspag@weborigin
```

7. Verify that the Media Flow Controller allows the trusted user to log in using the known host without using a password.

```
aspag@weborigin:~$ ssh aspag@10.100.100.100
Juniper Networks Media Flow Controller

Last login: Mon Dec  5 17:15:26 2011 from 10.13.111.10


Juniper Networks Media Flow Controller


LBS-NGEMFC2-1001 >
```

   According to the aboveinformation, the Media Flow Controller allows the trusted user to log in from the known host without a password.

Enter as many commands as needed after the syntax.

Related Documentation

- Setting SSH Keys for Multiple Hosts on page 26
- Before You Configure Media Flow Controller on page 20
- Understanding Media Flow Controller on page 5

Setting SSH Keys for Multiple Hosts

To set up SSH keys so that multiple machines can securely communicate:

1. Take the RSA key from the target host to which you want to push the logs and add it to Media Flow Controller. You can get the RSA host key from the following location in Linux:

   /etc/ssh/ssh_host_rsa_key.pub

2. Add the RSA key value to Media Flow Controller for the trusted user from the known host.

   (config) **# ssh client user <*username*> authorized-key sshv2 "<*host-key*>"**

   The quotation marks are required.

   The *host-key* is the entire host key entry in the RSA host key file that starts with **ssh-rsa** and ends with **== *trusted-username*@*hostname***.

3. Generate the public and private keys in Media Flow Controller.

   (config) # ssh client user admin identity rsa2 generate

4. Obtain the Media Flow Controller public key.

   (config) # show ssh client
   SSH client Strict Hostkey Checking: ask
   No SSH global known hosts configured.
   User Identities:
   User admin:
   RSAv2 Public key:
   ssh-rsa
   AAAAB3NzaC1yc2EAAAABIwAAAIEAzsG6puPiaHPrvFp8oTCiiL+JyqdQ2h792sjADfcflq6qK
   jq/
   Tq8pCJM2Hy5R+8vvAIRWASGBMOypaEYE3liUHUhD+vGUChbgJGRh7HlURclC3Sy+JtWkH45ox
   KKsuaSmlaPLGlH8KU3jeQ9XDMc4BR293vq5q4FmvI2MJkkCfb0=

5. Paste the Media Flow Controller public key in the following path on the machine where you want to upload the file.

   "/user_name/.ssh/authorized_keys"

6. To verify, do a manual upload.

   SFTP

   (config) # upload accesslog current sftp://user@host:path

For example:

sftp://root@192.168.1.10:/tmp

The first time you'll see something like this:

```
MFC (config) # upload accesslog current sftp://root@10.168.1.10:/tmp
sftp> cd /tmp
sftp>  put access2.log access2.log.tmp
Uploading access2.log to /tmp/access2.log.tmp
sftp> -rm access2.log
Couldn't stat remote file: No such file or directory Removing /tmp/ access2.log
Couldn't delete file: No such file or directory
sftp> rename access2.log.tmp access2.log
sftp>  exit
MFC (config) #
```

SCP

When using SCP protocol for uploading files, you may be prompted for a password. Press Enter if log in using keys is enabled (for example, if **password-less** is enabled).

```
MFC (config) # upload accesslog default all scp://user@host<upload directory>
Password:
Uploading ... access.log_0_2013_03_10_03_09_16
MFC  (config) #
```

Related Documentation

- Using SSH in Automated Scripts (CLI) on page 24
- Before You Configure Media Flow Controller on page 20
- Understanding Media Flow Controller on page 5

Media Flow Controller System Configuration Overview (CLI)

You can configure many basic system settings using the Media Flow Controller CLI.

NOTE: Some settings, such as the terminal length and width, are session-specific, and have no corresponding commands to set defaults. Also, some commands are only available in default form.

Save your settings after each configuration with the **write memory** command.

To configure system settings:

- Configure interfaces, the hostname, domain list, DNS, and default gateway.
  See "Configuring Interfaces, Hostname, Domain List, DNS, and Default Gateway (CLI)" on page 28 and"Example: Media Flow Controller Interface Configuration" on page 39.
- Configure the system clock and banner.
  See "Configuring Media Flow Controller Clock and Banners (CLI)" on page 43.
- Configure link bonding and static routes.
  See "Creating and Configuring Link Bonding and Static Routes (CLI)" on page 44.
- Understand authentication, authorization, and user account options.
  See "Understanding Authentication, Authorization, and User Options" on page 46.
- Configure user accounts.
  See "Configuring Media Flow Controller User Accounts (CLI)" on page 47.
- Apply the Media Flow Controller license.
  See "Applying the Media Flow Controller License (CLI)" on page 48.

Related Documentation

- Media Flow Controller Logging Overview on page 332
- Media Flow Controller Load Balancing Overview on page 193

Configuring Interfaces, Hostname, Domain List, DNS, and Default Gateway (CLI)

Typical network connectivity configuration of the Media Flow Controller consists of configuring interfaces with IP addresses, the Domain Name System (DNS), and the default gateway. The domain list helps resolve unqualified hostnames. Before you configure Media Flow Controller interfaces, see "Before You Configure Media Flow Controller" on page 20. See **interface** and **ip** in the *Media Flow Controller CLI Command Reference* for CLI details.

TIP: You might temporarily increase the CLI default logout time, 900 seconds (15 minutes), to give yourself more time. To do this, use the following command:

```
cli session auto-logout seconds
```

To configure Media Flow Controller interfaces, hostname, domain list, DNS, and default gateway:

1. Configure interface IP addresses for management (Eth0) and origin fetch (Eth1). You will later use the **delivery protocol** commands to configure traffic interfaces as needed (described in "Media Flow Controller Policy Configurations Overview" on page 49).

   Use the **show interfaces** command to verify.

   interface *interface_name* ip address *management_IP_address netmask_or_length*

2. Configure the system hostname. Use the **show hosts** command to verify.

   hostname *name_for_the_appliance*

3. Enable or disable (with **no**) use of DHCP on the specified interface. When enabled, DHCP gets the IP address and netmask, so those settings are ignored. Conversely, setting the IP address and netmask disables DHCP implicitly. Use **renew** to force a restart on the DHCP client for the specified interface. The default is **disabled**.

   interface *interface_name* dhcp

4. Optionally, add an **alias** (a secondary address) for this interface and set an **index** (name) to create a pseudo-interface with an **ip address** and **netmask**. Use the **show interface** command to verify.

   interface *interface_name* alias *index* ip address
   IP_address netmask

5. Configure a domain list (to resolve unqualified hostnames), and name server (DNS). Use the **show hosts** command to verify.

   ip domain-list *domain_name_for_resolving_hostnames*...
   ip name-server *DNS_server_IP_address*

6. Configure the default gateway. Use the **show ip default-gateway** command to verify.

   ip default-gateway *default_gateway_IP_address*

7. Since delivery changes have been made, restart the delivery service (**mod-delivery**).

   service restart mod-delivery

8. If you have configured a virtual interface (such as a loopback interface) or made any configuration changes to a virtual interface, you need to restart the SSL (HTTPS) service when you want to use this interface for HTTPS media delivery. To restart the SSL service, use the **service restart mod-ssl** command.

   The CLI displays the following message when you configure such interfaces:
   If HTTPS media delivery service has to use this configuration change then please restart mod-ssl using the 'service restart' command.

   For example, you would restart the SSL service when you configure:

   cmbu-cl28 (config) # interface eth0 alias 1 ip address 20.1.1.1 /24
   Information: If HTTPS media delivery service has to use this configuration change then please restart mod-ssl using the 'service restart' command

NOTE: You can include or exclude a disk pre-read status before displaying the system status as UP by entering the command **service status mod-delivery {include | exclude} disk**. You can check the status of mod-delivery service by entering the command **show service mod-delivery**.

A sample configuration from the unconfigured login prompt follows:

```
mfc-unconfigured-8a4990 login: admin
mfc-unconfigured-8a4990 > enable
mfc-unconfigured-8a4990 # configure terminal
mfc-unconfigured-8a4990 (config) # interface eth0 ip address 123.45.10.9 /24
mfc-unconfigured-8a4990 (config) # hostname MFC
MFC (config) # interface sit0 dhcp
MFC (config) # interface sit0 alias sit0a 123.45.11.9 /24
MFC (config) # ip domain-list example.local
MFC (config) # ip name-server 172.19.172.1
MFC (config) # ip default-gateway 123.45.10.1
MFC (config) # service restart mod-delivery
```

NOTE: Additional interface commands, such as adding a **comment**, using **identify** to flash LED lights, and configuring the **txqueuelen** (transmit queue length), are not included in this configuration example. See **interface** in the *Media Flow Controller CLI Command Reference* for CLI details.

Cutting and Pasting an Interface Configuration (CLI)

You can copy this series of commands, plug in your variables, and save them to a file to reuse as needed.

```
enable
configure terminal
interface eth0 ip address IP_address (netmask | mask_length)
hostname name
ip domain-list domain_name_for_resolving_hostnames...
ip name-server IP_address
ip default-gateway IP_address
service restart mod-delivery
write memory
```

Related Documentation

- Before You Configure Media Flow Controller on page 20
- Media Flow Controller Policy Configurations Overview on page 49

Using DNS Name Resolver Overview

Use the **name-resolver** command to configure DNS resolution related settings in Media Flow Controller. Media Flow Controller uses the configured DNS server to resolve or convert a domain name to an IP address. Media Flow Controller performs domain name resolution in two ways: synchronously and asynchronously (the recommended default). For more detailed information about configuring and using the **name-resolver** command, see the *Media Flow Controller CLI Command Reference*.

Related Documentation

- Multiple IP Address Support in DNS Cache on page 31
- About Media Flow Controller on page 3
- Before You Configure Media Flow Controller on page 20
- Displaying DNS Cache Entries on page 33

Multiple IP Address Support in DNS Cache

When using only a single IP address to establish a connection with the origin server, you increase the risk of a connection failure. You can enable support of a list of multiple IP addresses to reduce the chances of a connection failure. Media Flow Controller can be configured to use a list of multiple IP addresses returned by the DNS server to establish a connection with the origin server. A maximum of three IP addresses are cached per domain name.

If the primary IP address is unreachable for any reason or the TCP connection fails to be established after the configured timeout limit is exceeded, Media Flow Controller tries the next IP address in the list. The primary IP address is marked internally for deletion and further lookups to the domain name cache are not presented with this first IP address; logically, this is the same as removing the entry from the DNS cache. If the connection to the secondary IP address fails, the secondary IP address is marked internally for deletion and further lookups to the domain name cache are not presented with the second IP address. The third IP address is then tried. When the connection to the third IP address fails, the third IP address is marked internally for deletion and further lookups to the domain name cache are not presented with the third IP address, and the Media Flow Controller returns an HTTP 504 (gateway timeout) error message.

Whenever the time-to-live (TTL) has expired, stale entries that are based on a specific time interval are periodically removed from the DNS cache. Only subsequent requests for the same domain name cache will trigger a fresh lookup.

Multiple IP address support for DNS query resolution must be enabled using the command-line interface (CLI), where a maximum of three IP addresses are supported per domain. The single, primary IP address support mode is used only if the multiple IP address support feature is disabled .

Use the CLI to configure the amount of time allowed for the network to connect to the origin server. If the amount of time allowed to make the connection is exceeded (the connection timeout value, in seconds), then Media Flow Controller tries the next IP address in the list.

NOTE: In the scenario where a TCP connection is already established but a timeout occurs during data transfer, an idle timeout error (504 + 50001) is triggered. In this case, read/write errors occur at the application layer and failover action does not occur.

You can also use the CLI to display the first 50 DNS cache entries or display them by their fully qualified domain names (FQDNs).

Related Documentation

- Enabling Multiple IP Address Support in DNS Cache on page 32
- Verifying Tthat Multiple IP Address Support in DNS Cache Is Enabled on page 33
- Displaying DNS Cache Entries on page 33
- Deleting DNS Cache Entries on page 34

Enabling Multiple IP Address Support in DNS Cache

You can enable support of a list of multiple IP addresses to reduce the chances of a connection failure. Media Flow Controller can be configured to use a list of multiple IP addresses returned by the DNS server to establish a connection with the origin server.

Enable Media Flow Controller multiple IP address support for DNS query resolution using the command-line interface (CLI). If you do not enable this feature, only the single, primary IP address is supported for DNS resolution.

You can configure the connection timeout value, which is the amount of time allowed for Media Flow Controller to establish a three-way TCP handshake with the origin server. The timeout value is set to 10 seconds by default.

To verify that multiple IP addresses are available, you can view all the DNS cache entries or view only those that are fully qualified domain names (FQDNs).

To enable Media Flow Controller multiple IP address support for DNS query resolution and set the origin server connection timeout value:

1. In configuration mode, enable multiple address support in the DNS cache.

   network connection origin failover use-dns-response

   Use the **no** form of this command to disable multiple IP address support in DNS cache (the default condition).

2. The default origin server connection timeout value is 10 seconds and must be within the range of 6 to 120 seconds. To change the timeout value to a value other than 10 seconds, enter the following command in configuration mode:

   network connection origin connect timeout *timeout*

   Use the **no** form of this command to reset the timeout value to the default value..

   If you use the **network connection origin connect timeout** command to configure the timeout value to be *n* seconds, and Media Flow Controller is unable to connect to the first IP address among the multiple IP addresses returned by the DNS server for a domain (for the next *n* seconds from the time it had sent the first SYN packet to it), then Media Flow Controller fails over to the next IP address associated with that domain after *n* to *n*+2 seconds.

Related Documentation

- Multiple IP Address Support in DNS Cache on page 31
- Verifying Tthat Multiple IP Address Support in DNS Cache Is Enabled on page 33
- Displaying DNS Cache Entries on page 33
- Deleting DNS Cache Entries on page 34

Verifying Tthat Multiple IP Address Support in DNS Cache Is Enabled

Purpose Verify that multiple IP address list support for resolution mode is enabled.

Action Run **show network** to verify multiple IP address list support.

```
mfc # show network

Network time out (seconds)                  : 60
Maximum concurrent sessions                 : 64000
Maximum origin queue delay                  : 3
Maximum origin queue number                 : 2
Origin server failover support              : Enabled
Origin server connection timeout value      : 10
Per Session Maximum bandwidth (Kbits/sec)   : 0
Bind Socket to Interface                    : no
Syn-Cookie                                  : Enabled
Syn-Cookie Half-Open-Connection             : 1024
Max-Orphan                                  : 131072
FIN-Timeout                                 : 20
MAX-Buckets                                 : 184320
TCP Memory[low, high, maxpage]              : 196608   262144  1572864
TCP Read Memory[min, default, max]          : 4096   87380  16777216
TCP Write Memory[min, default, max]         : 4096   65536  16777216
PATH-MTU discovery                          : Enabled
IP Forward                                  : Disabled
TXQUEUELEN of Interface eth0                : 4096
IP Tables Enabled                           : yes
IGMP Version                                : 2
mfc #
```

Meaning Verify that the multiple address list support is enabled by checking that the line **Origin server failover support: Enabled** appears in the output.

Related Documentation

- Multiple IP Address Support in DNS Cache on page 31
- Enabling Multiple IP Address Support in DNS Cache on page 32
- Displaying DNS Cache Entries on page 33
- Deleting DNS Cache Entries on page 34

Displaying DNS Cache Entries

Purpose Display all of the DNS cache entries or only a specific FQDN cache entry.

Action

1. To display all the DNS cache entries:

show name-resolver cache

2. To display the DNS cache entry for a specific domain:

   show name-resolver cache *fqdn*

Meaning A list of all DNS cache entries or the specified FQDN cache entries are displayed.

Related Documentation

- Multiple IP Address Support in DNS Cache on page 31
- Enabling Multiple IP Address Support in DNS Cache on page 32
- Verifying Tthat Multiple IP Address Support in DNS Cache Is Enabled on page 33
- Deleting DNS Cache Entries on page 34

Deleting DNS Cache Entries

In configuration mode, to delete DNS cache entries.

1. To delete all DNS cache entries:

   name-relolver cache-delete all

2. To delete a specific domain DNS cache entry:

   name-resolver cache-delete *fqdn*

Related Documentation

- Multiple IP Address Support in DNS Cache on page 31
- Enabling Multiple IP Address Support in DNS Cache on page 32
- Verifying Tthat Multiple IP Address Support in DNS Cache Is Enabled on page 33
- Displaying DNS Cache Entries on page 33

Response-Based Cache Index Overview

Media Flow Controller uses a cache index to uniquely identify objects in the cache file system. In previous releases , Media Flow Controller found unique objects in cache by matching the static portion of a URI to the cache index and ignoring the dynamic part of the URI. Media Flow Controller generates the cache index by calculating a MD5 checksum from the HTTP response header, response data, or both. This feature is needed for sites that do not have a static portion in their HTTP Request URI. For example, a website might represent all video objects using the same URI; however, the objects are differentiated based on the ETag header. The administrator can use the response-based cache index (RBCI) feature for caching such websites.

Media Flow Controller can generate a cache index using:

- Response headers only (a maximum of eight response headers allowed in checksum computation).

- Response data only (1 through 32 KB from the beginning of the response data is taken for checksum computation).
- Response headers and response data.

Media Flow Controller uses the following response headers as a cache index: ETag, Last-Modified Header (LMH), and Content-Length. For more information about these response headers, see "Configuring Response-Based Cache Indexing" on page 37.

The following example shows the ETag, LMH, and Content-Length fields in an HTTP response and how two different URIs point to the same content because the ETag is the same:

```
http://f61.novamov.com/dl/12f34da035c6d3c2460ecb5a1c7b293d/4defb825
/7782a355b400e3835f035dc88176e151.flv

(Status-Line) HTTP/1.1 200 OK
Age 81
Cache-Control max-age=28800
Connection Keep-Alive
Content-Length 191900871
Content-Type video/x-flv
Date Wed, 08 Jun 2011 18:04:33 GMT
Etag "4193109138"
Expires Thu, 09 Jun 2011 02:04:33 GMT
Last-Modified Wed, 08 Jun 2011 14:16:52 GMT
Server vstreamer
Via 1.1 vxa22-5:80

http://f62.novamov.com/dl/84f79399919260c4d53721964ccfe2a3/4defb9c2/
7782a355b400e3835f035dc88176e151.flv
(Status-Line) HTTP/1.1 200 OK
Cache-Control max-age=28800
Connection Close
Content-Length 191900871
Content-Type video/x-flv
Date Wed, 08 Jun 2011 18:12:04 GMT
Etag "4193109138"
Expires Thu, 09 Jun 2011 02:12:04 GMT
Last-Modified Wed, 08 Jun 2011 14:16:52 GMT
Server vstreamer
Via 1.1 vxa22-5:80
```

The default cache index format is: **/<namespace_name>:<uuid>_<domain>:<port>/uri**.

For example, consider the original request URL: http://www.megavideo.com/file/5ff00d8cf918ac3626ec1d824a3a9fca. With RBCI configured to include an ETag header, from the response to the above request, **Etag: 12acef-gthjvk** is parsed. Finally the ETag-based cache index is: **/<namespace_name>:<uuid>_www.megavideo.com:80/file/51e6b557c2adf16cfc3944dc5fe22f14**.

Media Flow Controller response-based cache indexing (RCBI) supports byte-range based to an object stored in cache, which is indexed using RBCI. Media Flow Controller finds the correct object in cache by requesting the first part of the object from the origin, attempting to match that with the cache index of an object stored in cache) and delivers the appropriate byte-range from the cache. If the requested byte-range is not in the

cache, Media Flow Controller goes to the origin and fetches the appropriate byte-range, stores it in cache, and serves it to the user.

For example, if a range request comes from the user, requesting bytes 0 to 100 of a particular object, and if the administrator has configured 0 to 50 bytes to be used towards the checksum computation, Media Flow Controller creates a cache index out of the first 50 bytes of the object and caches the object with the RBCI derived from checksum.

If the range request from the user is for byte ranges 200 to 300, and if the administrator has configured 0 to 50 bytes to be used for checksum computation, Media Flow Controller fetches 0 to 50 bytes of the object, computes the checksum, and uses it for RBCI. If there is no cache hit, Media Flow Controller fetches byte range 200-300 from the origin, serves it to the user, and also caches the data from the byte range response.

To configure a response-based cache index, use the **namespace *name* delivery protocol http cache-index header include *Header name*** CLI command. To configure a response object cache index, use the **namespace mytestspace delivery protocol http cache-index object include checksum <number 1 to 32768>** command. For more information, see the *Media Flow Controller CLI Command Reference*.

NOTE: In a cluster proxy tier 1 configuration, if RBCI failed to generate the cache-index, the incoming request is not tunneled even when the dynamic-uri-no match tunnel is configured.

Related Documentation

- Configuring Response-Based Cache Indexing on page 37
- Response-Based Cache-Index Counters on page 39

Configuring Response-Based Cache Indexing

Response-based cache indexing (RBCI) uses the value of response headers to create cache indexes. Two successive requests are sent to the server: the first request to get all the response header values from origin, and the second request to do a local cache lookup.

You configure RBCI with byte-range support to:

- Include the checksum of the response header values in the cache index. The generated checksum is 16 bytes.
- Include the checksum of the first *n* bytes of the payload in the cache index. The generated checksum is 16 bytes.
- Include the checksum of the response header and response data. The generated checksum is 32 bytes (that is, 16 bytes + 16 bytes).

NOTE: **You can configure only one response header cache-index expression per namespace instance. The expression includes a cache-index header and object.**

To configure RCBI to include the response header values in the cache index:

1. Enter CLI configuration mode.

   > **enable**

   # **configure terminal**

2. Include the response header values in the cache index.

   (config) # **namespace *name* delivery protocol http cache-index headers include *Header name***

   For example:

   namespace megavideo delivery protocol http cache-index headers include etag
   namespace megavideo delivery protocol http cache-index headers include content-length

   You can include 1 to 8 headers whose values are added to a cache-index name.

   If the response has an ETag value of "12acef-gthjvk" and Content-LengthValue of 1024, then Media Flow Controller generates an MD5 checksum value (16 bytes) and the cache index would be: "//51e6b557c2adf16cfc3944dc5fe22f14" (16 byte MD5 checksum value).

   Examples of response headers values used to create a cache index include:

   - ETag—An optional, uniquely generated string in the HTTP response that identifies a piece of content.

- Last Modified Header (LMH)—Indicates when an object was modified, along with the content length and URI name to determine whether different URIs point to the same object.
- Content-Length—Indicates the size of the entity body or the transfer length of the message body.

To exclude the response header values in the cache index, use the **namespace *name* delivery protocol http no cache-index headers include *Header name*** CLI command.

3. Include the checksum of the first *n* bytes of the payload in the cache index.

   Inclusion of the object checksum in the cache index helps to ensure that the correct object is served to the user.

   (config) # **namespace *name* delivery protocol http cache-index object include checksum *<number 1 to 32768>***

   For example:

   ```
   namespace megavideo delivery protocol http cache-index object include checksum
   1024
   ```

   If the response has an ETag value of "12acef-gthjvk" and the 1K data checksum value is 13413182a280f55f4e1b8fcae2416bcb (16 bytes), then our internal cache index would be: '/12acef-gthjvk_13413182a280f55f4e1b8fcae2416bcb'

   To exclude the object checksum in the cache index, use the **namespace *name* delivery protocol http no cache-index object include checksum *<number 1 to 32768>*** CLI command.

   NOTE: If you configure Media Flow Controller to generate a cache index on the basis of the response headers and response data, then Media Flow Controller appends each individually calculated checksum value for response headers (16 bytes) and response data (16 bytes) to generate the 32-byte MD5 checksum value for the cache index.

4. View the namespace configuration.

   (config) # **show run**

   ```
   ## Namespace Configuration
   ##
   . . .
   namespace test delivery protocol http cache-index header include "etag"
   ```

```
namespace test delivery protocol http cache-index object include checksum 25
...
```

1. Enter CLI configuration mode.

   > **enable**

   # **configure terminal**

Related Documentation

- Response-Based Cache Index Overview on page 34
- Response-Based Cache-Index Counters on page 39

Response-Based Cache-Index Counters

Table 4 on page 39 displays the response-based cache indexing counters that provide debugging and statistics support.

Table 4: Response-Based Cache-Index Counters

Counter	Description
glob_resp_cache_index_no_header_err	Error condition where the configured response header is not present in the reply from the origin server.
glob_resp_cache_index_header_parse_err	Error condition where the configured response header from the origin server results in a parse error.
glob_resp_cache_index_cod_open_err	Error condition when the cache-index-based cod open fails.
glob_resp_cache_index_tunnel_x (x ranges from 1-10)	Conditions while in response cache-index path where tunnel path is taken.
glob_resp_cache_index_tot_requests	Gathers the total number of cache index-based requests.
glob_resp_cache_index_cur_requests	Shows the number of active cache index-based requests.
glob_resp_cache_index_dyn_uri_cnt	Shows the total number of cache-index plus dynamic URI combinations.
glob_resp_cache_index_dyn_uri_ssp_cnt	Shows the total number of cache-index plus dynamic URI plus SSP combinations.

Related Documentation

- Response-Based Cache Index Overview on page 34
- Configuring Response-Based Cache Indexing on page 37

Example: Media Flow Controller Interface Configuration

When Media Flow Controller initializes, the on-board Ethernet interfaces are numbered eth0, eth1, and so on. When a NIC, dual-port or quad-port, is attached to the server, the first NIC (by PCI channel number) is assigned interface names eth10, eth11, and so on, to

eth19. The second NIC is assigned the names eth20, eth21, and so on, to eth29; assuming that only up to 10 Ethernet interfaces exist per NIC. See Table 5 on page 40, for details.

CAUTION: For VXA Series Media Flow Engine appliances, never change the Ethernet name mappings; all interface assignments are handled automatically during manufacturing.

In Table 5 on page 40, and Figure 4 on page 41, for example, the wiring logic is as follows:

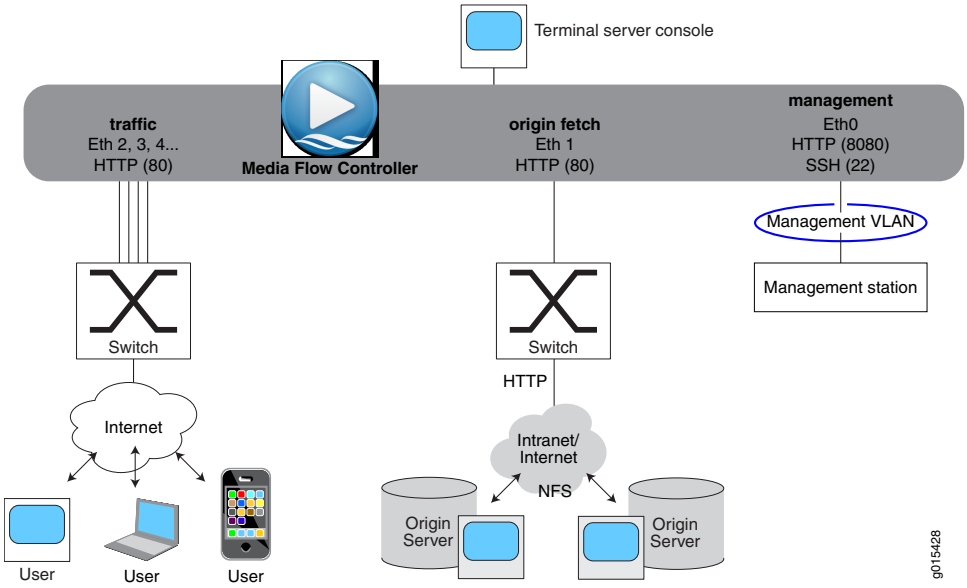
- **eth0** —Running SNMP, sending analytics to another machine, Web management, SSH, and Telnet; connected to your internal network.
- **eth1**—Upstream fetching content from origin; connected to the network that connects to the origin server.
- **eth2 through eth5** —Service traffic; connected to the public Internet. These interfaces must have IP addresses.

Table 5: Example Machine Setup of Management and Traffic Ports

Interface	Connectivity	IP Address	Subnet Mask	Open Ports	Internet Access	Purpose
eth0	onboard	192.168.1.100	255.255.255.0	8080, 22	yes	Management
eth1	onboard	172.20.46.10*	255.255.255.0	80	yes	Origin fetch
eth2	PCIe card	10.1.1.11	255.255.255.0	80	not applicable	Traffic
eth3	PCIe card	10.1.2.11	255.255.255.0	80	not applicable	Traffic
eth4	PCIe card	10.1.3.11	255.255.255.0	80	not applicable	Traffic
eth5	PCIe card	10.1.4.11	255.255.255.0	80	not applicable	Traffic

*eth0 AND eth1 can be on the same subnet; this examples indicates they are not.

Figure 4: Example Connectivity



eth0 and eth1, typically the first two interfaces you use, are usually the first two network ports built into the system—either part of the system board, or the first add-in card or module.

Media Flow Controller supports Lights Out (or "out-of-band") management, which involves the use of a dedicated management channel for device maintenance. This lets you monitor and manage servers and other network equipment by remote control regardless of whether the machine is powered on. You can configure out-of-band management through the BIOS.

NOTE: Media Flow Controller does not support RAID arrays.

Related Documentation

- Before You Configure Media Flow Controller on page 20
- Configuring Network Connectivity for IPv6 (CLI) on page 41
- Media Flow Controller Policy Configurations Overview on page 49

Configuring Network Connectivity for IPv6 (CLI)

Typical network connectivity configuration of the Media Flow Controller consists of configuring interfaces with IP addresses and default gateway for IPv6. Before you configure Media Flow Controller interfaces, see "Before You Configure Media Flow Controller" on page 20. See **interface** and **ipv6** in the *Media Flow Controller CLI Command Reference* for CLI details.

TIP: You may want to temporarily increase the CLI default logout time, 900 seconds (15 minutes), to give yourself more time. To do this, use this command:

cli session auto-logout *seconds*

To configure Media Flow Controller interfaces, hostname, default gateway, and static routes:

1. Configure interface IP addresses for management (Eth0), and origin fetch (Eth1). You will later use the **delivery protocol** commands to configure traffic interfaces as needed (described in "Media Flow Controller Policy Configurations Overview" on page 49).

   Use the **show interfaces** command to verify.

   interface *interface_name* ipv6 address (*IP_address/length* | autoconfig)

2. Enable IPv6 on this interface. Use the **show interfaces** command to verify.

   interface *interface_name* ipv6 enable

3. Configure the system hostname. Use the **show hosts** command to verify.

   hostname *name_for_the_appliance*

4. Configure the default gateway. Use the **show ipv6 default-gateway** command to verify.

   ipv6 default-gateway *default_gateway_IP_address*

5. Configure the static route used to route data from one subnet to another. Use the **show ipv6 route static** command to verify.

   ipv6 route *network_prefix* [*next_hop_IP_address* | *interface_name* [*interface_name*]]

6. Since delivery changes have been made, restart the delivery service (**mod-delivery**).

   service restart mod-delivery

A sample configuration from the unconfigured login prompt follows:

```
mfc-unconfigured-8a4990 login: admin
mfc-unconfigured-8a4990 > enable
mfc-unconfigured-8a4990 # configure terminal
mfc-unconfigured-8a4990 (config) # interface eth0 ipv6 address 2001::13/64
mfc-unconfigured-8a4990 (config) # interface eth0 ipv6 enable
mfc-unconfigured-8a4990 (config) # hostname MFC
MFC (config) # ipv6 default-gateway eth0
MFC (config) # ipv6 route 2001::/64
MFC (config) # service restart mod-delivery
```

NOTE: Additional interface commands are not included in this configuration example. See **interface** in the *Media Flow Controller CLI Command Reference* for CLI details.

NOTE: The Media Flow Controller Web user interface does not display IPv6 addressing related information.

Related Documentation

- Before You Configure Media Flow Controller on page 20
- Media Flow Controller Policy Configurations Overview on page 49

Configuring Media Flow Controller Clock and Banners (CLI)

Proper time configuration is required for accurate functioning. A network time protocol (NTP) server, or a group of NTP servers (peers) is desirable. Banners allow you to display messages to users. Before you configure the Media Flow Controller system clock and banners, see "Before You Configure Media Flow Controller" on page 20. NTP is enabled by default.

To configure an NTP (network time protocol) server OR system clock and time zone:

1. Configure the NTP server. Use the **show ntp** command to verify.

   ntp server *IP_address*

   Alternatively, configure the system clock, and time zone. Use the **show clock** command to verify.

   clock set *hh:mm:ss* [*yyyy/mm/dd*]
   clock timezone *zone* [*zone_word*] [*zone_word*] ...

2. Optionally, configure the NTP peers. If you do not specify version number **3**, version **4**, the default, is used.

   ntp peer *IP_address*

3. Optionally, configure banners. There are two configurable banners: **motd** (message of the day) and **login**. In the CLI, both are displayed at the command line when you log in. In the Management Console, only the **login** message appears on the login page. Multi-word messages must be surrounded by quotes. Use the **show host** and **show banner** commands to verify.

   banner [login *message_string*] [banner motd *message_string*]

Example configuration:

```
MFC (config) # ntp server 123.45.10.7
MFC (config) # ntp peer 123.45.10.8
MFC (config) # clock set 15:51:30
MFC (config) # clock timezone America North United_States Pacific
MFC (config) # banner login “Welcome to Media Flow Controller”
MFC (config) # banner motd “Please note new link bonding commands”
```

Related Documentation

- Before You Configure Media Flow Controller on page 20
- Creating and Configuring Link Bonding and Static Routes (CLI) on page 44
- About the Media Flow Controller CLI on page 21

Creating and Configuring Link Bonding and Static Routes (CLI)

Link bonding allows you to increase link speed by using multiple ports simultaneously. Static routes can be useful for the "stub" parts of your network that need only connect to each other. Before you configure Media Flow Controller link bonding and static routes, see "Before You Configure Media Flow Controller" on page 20.

Media Flow Controller supports three bonding modes:

- **balance-rr**—Round-robin mode sends TCP/IP packets belonging to the same session across multiple links. Out-of-order TCP packets coming through different links are retransmitted. This mode supports load balancing and failover.
- **balance-xor-layer3+4**—Traffic to a particular network peer goes across multiple links, although packets belonging to a single connection or session do not span multiple links. This mode supports load balancing and failover. Link selection is based on the TCP port and IP address.
- **link-agg-layer3+4**—Link Aggregation Control Protocol (LACP) allows the automatic negotiation of port bundling to form a single logical channel between LACP-enabled links. This mode supports load balancing and failover.

You bond interfaces to create a port channel or aggregated link for load distribution across links and increased link availability. Example shows bonding interfaces Eth10 and Eth11 as a named bonded interface " 0" . In this way, Layer 2 packets are distributed across the defined links for load distribution; if one link fails, the other links take over the media delivery. After you have created the bonded interface, you can use the **delivery protocol** command to assign the interface as a traffic interface and configure its listen port, if needed.

Individual links under a bond of 1 Gig interfaces cannot be configured for different speeds (10/100/1000). Only one bond interface is allowed.

NOTE: **Do not name a bond interface " 0" or " 1" as these names can cause problems with the TCP dump function. It is better to use alphanumeric characters such as " ae0," " b1," or " bond0".**

To configure link bonding and static routes:

1. Create the bond interface with a **name** and specify a **mode**.

   The CLI lists several options for **bond** ***bond_interface*** **mode** that are not supported. Only **balance-rr** (round-robin), **balance-xor-layer3+4** (non-LACP), and **link-agg-layer3+4** (LACP) are supported.

   bond *name_for_virtual_interface* mode *bond_mode*

2. Add up to four interfaces to bond. Mixed mode bonding (1 Gig and 10 Gig interfaces bonded in a single bond) is not supported.

   interface *interface_name virtual_interface_name*

3. Assign the new bonded interface as a traffic interface and set non-default (80) listen ports, if needed; up to 64 ports can be assigned. After you assign a traffic interface, Media Flow Controller accepts traffic only on those assigned interfaces (up to 10). By default, Media Flow Controller accepts traffic on all interfaces.

   delivery protocol http interface *bonded_interface_name*
   delivery protocol http listen port *port port port*

   Example:

   ```
   MFC (config) # bond b1 mode balance-rr
   MFC (config) # interface eth10 bond b1
   MFC (config) # interface eth11 bond b1
   MFC (config) # delivery protocol http interface bond b1 eth12 eth13
   MFC (config) # delivery protocol http listen port 80 81 82
   MFC (config) # show bonds
   Bonded Interface b1:
     Enabled: yes
     Mode: balance-rr
     Link Monitor Time: 100
     Interfaces:
       eth10
       eth11
   ```

4. Configure the IP address for the bonded interface.

   MFC (config) # interface b1 *IP_address*

5. Optionally, configure static routes and ensure a static host mapping for the defined hostname. The **ip route** command only works on devices that already have an IP address assigned. Use the **show ip route** command to verify.

   ip route *network_prefix* (*netmask* | *mask_length*) (*next_hop_IP_address* | *interface_name*)
   ip map-hostname

   Example:

   MFC (config) # ip route 123.45.10.0 /24 eth0

6. Since delivery changes have been made, restart the delivery service.

   service restart mod-delivery

NOTE: Bonded interfaces show Speed and Duplex as UNKNOWN in the **show interfaces** command output; this is not an error condition.

Related Documentation

- Before You Configure Media Flow Controller on page 20
- Load Balancing with Direct Server Return on page 194

Understanding Authentication, Authorization, and User Options

Several configurations or tasks can use an already configured authentication or authorization scheme (AAA, namespace pre-staging, users, file transfers, and so forth). Because authentication schemes can be complex to configure, this section does not attempt to guide you through the configuration steps for setting authentication or AAA options. However, it does provide references to the CLI commands. Before configuring any authentication or authorization schemes, you must know the hostname or IP address of the authenticating server, and the shared secret for authentication.

About MD5, SHA-1, AES-128, and DES

The first two algorithms, **MD5** and **SHA-1**, are cryptographic hash algorithms:

- **MD5**—Message Digest 5. Considered somewhat faster but less secure than sha1, but still supported for legacy systems. Generates a 128-bit (16 byte) hash.
- **SHA-1**—Secure Hash Algorithm 1. Considered more secure than MD5 but still vulnerable to collision attacks. Generates a 160-bit (20-byte) hash.

The second two, **AES-128** and **DES**, are encryption standards used to encrypt and un-encyrpt data.

- **AES-128**—Advanced Encryption Standard; 128 is a specific "block cipher." AES is a newer standard than DES and considered much more secure. Generates a 128-bit encryption key. AES is an asymmetric encryption algorithm, which means that the sender uses the public key of the receiver to encrypt the message and the receiver uses its private key to decrypt the message.
- **DES**—Data Encryption Standard. This standard is older than AES-128 and considered less secure than AES-128 but is still supported for legacy systems. DES generates a 56-bit encryption key. DES is a symmetric encryption algorithm, which means that you use the same key to encrypt and decrypt the message.

User Account Defaults and States

The system comes initially with four accounts already created:

- **admin**—Full privileges to do anything on the system.
- **mfc_probe_ftpuser**—The auto-created user for the internal watchdog.
- **cmcrendv**—Not Supported.
- **monitor**—Privileges to read almost everything on the system (excluding logs), and perform some actions, but cannot modify configurations.

These accounts are enabled, and by default no password is required for login.

An account may be in one of the following states:

- Account disabled (not listed in /etc/passwd—The **admin** account cannot be disabled.

  username foo disable

- Local password login disabled (hashed password set to "*")—There is no locally-configured password to permit the user to log in. The user may still log in using an SSH authorized key if one is installed, or using remote authentication (for example, RADIUS or TACACS+). The **admin** account cannot be in this state unless it has an SSH authorized key installed.

  username foo disable password

- All password login disabled (hashed password set to "!!")—No CLI command for this; the hashed password must be set to " !!" . Same as "Local password login disabled" except that the user cannot be remotely authenticated (for example, by a RADIUS or TACACS+ server). The user can still log in using an SSH authorized key if one is installed. The **admin** account cannot be in this state unless it has an SSH authorized key installed.

- Local password set—The user can log in by typing the password that has a hashed version stored. This is not necessary if an SSH authorized key is installed, or if a remote auth server comes earlier in the authentication order.

  username foo nopassword mypassword

- No password required for login (hashed password set to "")—Anyone can log in to this account without providing authentication. The **admin** and **monitor** accounts begin in this state (unless overridden by configured defaults), but should be changed for better security.

  username foo nopassword

Related Documentation

- Before You Configure Media Flow Controller on page 20

Configuring Media Flow Controller User Accounts (CLI)

Configure user accounts to allow multiple administrators to make configuration changes, or to allow certain users to view or monitor the appliance. Before you configure Media Flow Controller user accounts, see "Before You Configure Media Flow Controller" on page 20.

To configure users with the CLI:

1. Configure authentication or authorization.

   See **radius-server** and **tacacs-server** for CLI details.

2. After your authentication settings are made, configure authentication and authorization parameters such as setting the default login authentication order and default authorization mapping for local and remote users.

   See **aaa** for CLI details.

3. Configure users.

Media Flow Controller provides three capability sets for users: **admin** (full privileges), **monitor** (can view configurations but make no changes), and **unpriv** (very limited command access); see **username** for CLI details. In addition to these capabilities, you can configure password options and disable a user account. Use the **show usernames** command to verify.

a. Add a user and specify the capability; users are added with admin privileges:

   username *username* capability *capability*

b. Delete a user:

   no username *username*

c. Disable a user's password.

   This command does not remove the user or the password:

   username *username* disable password

d. For a defined user, allow no password:

   username *username* nopassword

e. For a defined user, configure a password.

   If no password is specified, the user logs in with no password. If **0** is specified, enter a password in cleartext (the system encrypts it using the DES algorithm) and the user logs in with that password. If **7** is specified, you must enter the previously-created, DES encrypted password for that user at the command line. The Media Flow Controller default **admin** user does not have a default password; set an **admin** password to secure and restrict administration.

   username password (0 *cleartext_password* | 7 *encrypted_password* | *cleartext_password*)

Example:

```
MFC (config) # username joe capability unpriv
MFC (config) # username joe password 12345
MFC (config) # username joe disable password
MFC (config) # username joe nopassword
```

Related Documentation

- Before You Configure Media Flow Controller on page 20

Applying the Media Flow Controller License (CLI)

Media Flow Controller comes unlicensed, and by default, can only support 10 connections at 200 Kbps each. Neither the number of sessions nor the session rate (network connection parameters) is configurable without the Media Flow Controller license. Contact Juniper Networks to obtain the Media Flow Controller license for normal operations. You need to provide the host ID. Use the **show version** command to display the host id. Based on this information, Juniper Networks will provide you a license key. After installing the license, you get full feature capability. See **license** in the *Media Flow Controller CLI Command Reference* for CLI details.

To apply the Media Flow Controller license:

1. Install a license.

   license install *license_key*

2. Delete a license.

   license delete *license_key*

3. View installed licenses, including expiration dates.

   show licenses

   Example:

```
MFC (config) # show license
No licenses have been configured.
MFC (config) # show network
Network time out (seconds) : 60
Maximum concurrent sessions : 10
Per Session Maximum bandwidth (Kbits/sec) : 200
MFC (config) # license install LK2-MFC-413E-5N42-3EE6-4381-GLL8-CE98
MFC (config) # show license
License 1: LK2-MFC-413E-5N42-3EE6-4381-GLL8-CE98
  Feature: Media Flow Controller
  Valid: yes
  Start date: 2009/03/15 (ok)
  End date: 2009/06/30 (ok)
  Tied to MAC addr: 00:1E:C9:FF:0C:FA (ok)
  Active: yes
MFC (config) # show network
Network time out (seconds) : 60
Maximum concurrent sessions : 64000
Per Session Maximum bandwidth (Kbits/sec) : 0
```

Related Documentation

- Before You Configure Media Flow Controller on page 20
- Configuring Media Flow Controller User Accounts (CLI) on page 47

Media Flow Controller Policy Configurations Overview

After your appliance network connections and basic settings are configured, you are ready to start configuring Media Flow Controller policy settings. Any time network configuration changes are made, you must restart the delivery service (**mod-delivery**) using the **service restart mod-delivery** command. This includes initial configurations after installation. In the Management Console, you do this on the **EZconfig** page.

To configure policy settings:

- Set cache analytics options.
  See"Setting Analytics Options (CLI)" on page 50
- Set network connection options.
  See "Setting Network Connection Options (CLI)" on page 52

- Configure delivery options.
  See "Configuring Media Flow Controller Delivery Protocols (CLI)" on page 54
- Configure **resource-pool** options.
  See "Creating Resource Pools (CLI)" on page 56
- Create and configure virtual players.
  See "Virtual Players Overview" on page 117
- Create and configure namespaces.
  See "Creating a Namespace and Setting Namespace Options (CLI)" on page 76
- Manage the disk cache.
  See "Managing the Media Flow Controller Disk Cache (CLI)" on page 60

Related Documentation

- Media Flow Controller Logging Overview on page 332
- Media Flow Controller Load Balancing Overview on page 193
- Media Flow Controller Deployment Guidelines on page 447

Setting Analytics Options (CLI)

Setting Analytics Options (CLI)

Configure options for handling the promotion of objects. Use the **analytics** commands to set a **size-threshold** limiting, by size, what objects are promoted to faster tiers. You can set controls on **cache-promotion**, including disabling it entirely, or adjusting the hotness threshold for promoting objects to another cache tier. You can set a **memory-limit** on the amount of cache memory that can be given to storing header data. See **analytics** in the *Media Flow Controller CLI Command Reference* for CLI details. Before you configure Media Flow Controller cache **analytics** options, see "Before You Configure Media Flow Controller" on page 20.

- Media Flow Controller uses the concept of " hotness" to determine when to promote an object to various cache tiers. Tiers are comprised of RAM, SSD, SAS, and SATA. The hotness computation is based on the hit frequency (such as, the number of hits as a function of time). For example, an object with 400 hits an hour is considered hotter than an object with 500 hits in one day.

  The default and recommended setting for hotness threshold is **3**, for following reasons:

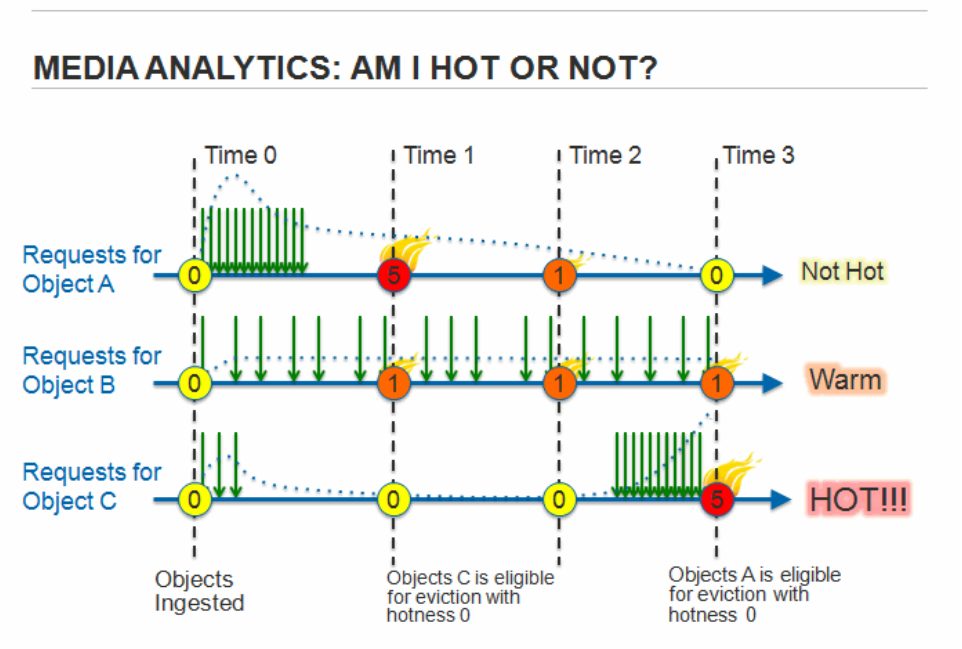
  - The analytics manager only keeps track of 200,000 objects; billions of objects will be stored in a typical transparent proxy deployment.
  - The hotness algorithm does not take object size into account and can promote a large hot object into the SSD, thereby causing a large number of smaller objects to be evicted and the SSD to be underutilized.

If necessary, use the following command to reset the default value:

namespace test delivery protocol http origin-fetch cache-ingest hotness-threshold 3

The higher the hotness threshold value is, the slower the object will get moved to a faster cache tier.

Figure 5: Object Hotness



Configuring Caching Analytics (CLI)

To configure caching analytics with the CLI:

1. Configure a **cache-tier highest object-size**, limiting which objects can be cached. Objects smaller than, or equal to, the configured size are automatically written to the fastest cache tier. The default is **0** (zero), no objects are directly promoted to the fastest tier. The maximum allowed value is **4294967295** (4 GB).

   namespace name1 delivery protocol http origin-fetch content-store media cache-tier highest object-size *<value>*

2. Configure cache-ingest hotness-threshold at the namespace level. The hotness value that an object should attain to get ingested to the slowest tier. The default value is **3**, which is the lowest possible hotness value for an object to get ingested in the first hit.

   namespace name1 delivery protocol http origin-fetch cache-ingest hotness-threshold

3. Disable or re-enable **cache-promotion** at the global level. The default is enabled; if **disable** is used, no cache promotion occurs.

   analytics cache-promotion (disable | enable)

4. Configure a cache **memory-limit** for analytics data. Use either a value between **0** and **2048**, in storage MiB (1,048,576); or set to this value **auto** to use the built-in algorithm. The default is **500** MiB.

   analytics memory-limit *MiB*

Related Documentation

- Before You Configure Media Flow Controller on page 20
- Caching and Origin Clustering on page 12

Setting Network Connection Options (CLI)

Configure network connection settings (global Media Flow Controller defaults) to limit and control connections. The **max-bandwidth** option is available in **virtual-player** configurations, which override **network connection** configurations. Before you configure Media Flow Controller **network connection** options, see "Before You Configure Media Flow Controller" on page 20.

- Configuring Network Connections (CLI) on page 52

Configuring Network Connections (CLI)

To configure network connections with the CLI:

1. Configure the global network concurrent session limit; the default is **10** (without a Media Flow Controller license), or **64000** (with a Media Flow Controller license). The maximum limit is **524,288** in Release 12.2 onward..

   network connection concurrent session (64000 | *integer*)

2. Configure the global network socket idled-out time, in seconds. This is the time the network waits before closing a connection when there is no data in session; the default is **60** seconds.

   network connection idle timeout *seconds*

   If you configure the timeout value using the **network connection idle timeout** command to be *n* seconds, and if the client connection goes into an idle state after establishing a connection with the Media Flow Controller or after receiving a response for its request, then Media Flow Controller closes the connection after *n* to *n*+2 seconds of idleness.

3. Configure the global network maximum allowed bandwidth (burst rate); even if there is available bandwidth in the link, Media Flow Controller does not allocate more than this value for a session. When there is a full download, Media Flow Controller tries to allocate this value to the session. The default is **200** without a Media Flow Controller license, **0** kbps (unbounded) with a Media Flow Controller license.

   network connection max-bandwidth (0 | *kbps*)

4. Verify the configurations.

   show network

5. Save the configuration settings.

   write memory

   Example:

   ```
   MFC (config) # network connection concurrent session 64000
   MFC (config) # network connection idle timeout 900
   MFC (config) # network connection max-bandwidth 2000
   MFC (config) # show network
   ```

```
Network time out (seconds): 900
Maximum concurrent sessions: 64000
Per Session Maximum bandwidth (Kbits/sec): 2000
Network Access-Control PERMIT list: NONE
Network Access-Control DENY list: NONE
```

Related Documentation

- Before You Configure Media Flow Controller on page 20
- Protecting Proxy Servers from Denial of Service Attacks on page 53

Protecting Proxy Servers from Denial of Service Attacks

A denial of service (DoS) attack can be carried out on an origin server from one or more hosts making many requests to fetch content from that origin server at a high enough rate that the origin server is not able to respond to requests fast enough. A DoS attack causes the origin server to become slow or to stop responding to connections altogether. When a Media Flow Controller is caching on behalf of this origin server, and these requests are seen by the Media Flow Controller as non-cacheable, it sends these requests directly to the origin. When the origin becomes inundated with all of these requests and starts to respond slowly or not respond at all, all of these unresponsive connections from the Media Flow Controller to the unresponsive origin server consume Media Flow Controller resources and can in turn cause the Media Flow Controller to become unresponsive.

To prevent connection buildup, Media Flow Controller, keeps track of the number of unresponsive connections for each server against a maximum connections threshold that you set, and drops new client requests for the same server if the unresponsive connections are more than the configured threshold value. Newly opened connections in the origin path are treated as unresponsive connections until the first byte is received from the origin. By dropping the requests, Media Flow Controller uses resources effectively to serve the existing clients without delay.

In both transparent and reverse proxy deployments, you can configure a threshold value for the number of unresponsive connections that Media Flow Controller accepts from an origin using the **network connection origin max-unresponsive-conns *value*** CLI command. The **max-unresposive-conns** threshold is common to all servers.

Use the **show network** CLI command to view the configured maximum unresponsive connections threshold value.

To configure the maximum unresponsive connections threshold value:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Configure the maximum unresponsive connections threshold value.

   network connection origin max-unresponsive-conns *value*

   Where *value* is the threshold value you configure. The threshold range is from **0** to **4294967295**. Zero (0), the default, disables the command.

Whenever a new request goes to the origin, the unresponsive connections counter for the respective server is checked against the threshold value. If the threshold value is exceeded, the request is closed. If the threshold value is not reached, the unresponsive connections counter for the server is increased by one. Once Media Flow Controller receives the first byte from the server, the unresponsive connections counter for the server is decreased by one. If there is no response from the origin for that request, the unresponsive connections counter for the server is decreased by one while closing the connection in the time out.

3. Exit configure mode.

   # **exit**

4. View the maximum unresponsive connections threshold.

   > **show network**

```
cmbu-vxa20-7 (config) # show network
Network time out (seconds)                    : 60
Maximum concurrent sessions                   : 64000
Maximum origin queue delay                    : 3
Maximum origin queue number                   : 2

--> Maximum origin unresponsive connections      : 1000

Origin server failover support                : Disabled
Origin server connection timeout value        : 10
Per Session assured flow rate (Kbits/sec)     : 0
Per Session Maximum bandwidth (Kbits/sec)     : 0
. . .
```

Related Documentation

- Setting Network Connection Options (CLI) on page 52

Configuring Media Flow Controller Delivery Protocols (CLI)

Media ports on the Media Flow Controller receive and deliver media. These ports typically have Internet access, and should be connected with highest-quality cables. See Table 5 on page 40 for an example.

Use the **delivery protocol** command to specify what protocols to use for media delivery and manipulate headers; in Version **http** and **rtsp** are allowed values. By default, Media Flow Controller listens on all interfaces; if you set specific traffic interfaces, only those are used for traffic. See **delivery** in the *Media Flow Controller CLI Command Reference* for CLI details, and see "Media Flow Controller Delivery Methods" on page 9 for background information. Before you configure the Media Flow Controller **delivery protocol** options, see "Before You Configure Media Flow Controller" on page 20.

To configure delivery protocol options with the CLI:

1. Media Flow Controller always supports known HTTP methods (GET, POST, TRACE, CONNECT, OPTIONS, DELETE, and PUT). To add support for **http** request methods, use **allow-req** and specify up to 16 custom request methods. Use **all** to permit Media Flow Controller to tunnel any request. The default is **none**, only the known methods listed are allowed. Use **no** to remove the specified method or methods.

delivery protocol http allow-req all

2. Optionally, disable or enable connection pooling (enabled by default) and configure parameters; this is for delivery protocol **http** only.

   delivery protocol http conn-pool origin (enable | disable)

   a. Use **max-conn** to limit the maximum allowed pooled connection; the default is **4096**, the maximum allowed is **128000**.

   b. Use **timeout** to configure a pooled connection timeout; the default is **90** seconds, the maximum allowed is 86,4000 seconds (24 hours).

      If you configure the timeout value using the **delivery protocol http conn-pool origin timeout** command to be *n* seconds, and the origin-side connection in the origin connection pool is not used for the next *n* seconds since the last time it sent the client a request and received response for the same request, then Media Flow Controller closes the origin-side connection after *n* to *n*+2 seconds.

3. Optionally, configure interfaces for Media Flow Controller traffic; when configured, Media Flow Controller accepts traffic on those interfaces only. This applies to both **http** and **rtsp** delivery protocols. You can specify up to 10 space-separated interfaces.

   delivery protocol (http | rtsp) interface *interface*, *interface*...

   a. To configure interfaces for transparent proxy, set HTTP interfaces:

      delivery protocol http interface *interface*, *interface*...

   b. Enable the interfaces for transparent proxy:

      delivery protocol http transparent *interface*, *interface*... enable

4. Optionally, configure listen ports for the traffic interfaces as needed; the default is port **80** for **http**, port **554** for **rtsp**. By default, Media Flow Controller listens on port 80 for HTTP and on port 554 for RTSP on all interfaces.

   delivery protocol http listen port *port*

5. Optionally, set the maximum request length (domain + URI + Query Params + Headers), in characters per bytes, for incoming requests (**http** delivery protocol only). The default is **16384** bytes; the maximum allowed value is **32768**. Incoming requests with lengths exceeding the configured value are rejected.

   delivery protocol http req-length maximum *bytes*

6. Optionally, enable origin side revalidation requests. The default is **no**. HEAD revalidation requests are more efficient than GET revalidation requests; however, some content websites do not support HEAD requests.

   delivery protocol http revalidate-get

7. Since delivery changes have been made (Steps 3 and 4), restart the delivery service (**mod-delivery**).

   service restart mod-delivery

8. Verify the configurations.

   show delivery protocol http

9. Save the configuration settings.

   write memory

   Example:

```
MFC (config) # delivery protocol http conn-pool origin max-conn 128000
MFC (config) # delivery protocol http interface eth11 eth12 eth13
MFC (config) # delivery protocol http listen port 80
MFC (config) # delivery protocol http req-length maximum 23576
MFC (config) # delivery protocol http revalidate-get
MFC (config) # service restart mod-delivery
MFC (config) # write memory
```

Related Documentation

- Before You Configure Media Flow Controller on page 20

Creating Resource Pools (CLI)

Create resource-pools for multi-tenant management. A resource pool reserves from Media Flow Controller's global resources configured bandwidth and connection capabilities. Once you create a resource pool, you bind namespaces to it. All the namespaces bound to the resource pool share the configured resource limits for that pool. See **resource-pool** in the *Media Flow Controller CLI Command Reference* for CLI details. See "Media Flow Controller Multi-Tenancy Management" on page 15 for background information.

NOTE: In Release 12.1, resource-pool does not work in the tunnel path; for example, whenever a request is not served by Media Flow Controller from its cache, but is tunneled to the origin, resource-pool limits are not applied. Typically, Media Flow Controller tunnels non-cacheable requests to the origin.

In Media Flow Controller Release 12.1, resource limits can be defined and monitored for client sessions only; they are not applicable for origin connections.

- How Resource Pools Work on page 56
- Bandwidth Management on page 58
- Configurable Maximum Sessions on page 58
- Resource Monitoring on page 59
- Configuring Resource Pools (CLI) on page 59

How Resource Pools Work

Resource pools allow you to configure resource utilization limits for concurrent connections and bandwidth usage. Resource pools are limited by the global resources available in the device. Globally available system resources are considered to be part of the "global resource pool." The global resource pool is the allowed maximum concurrent sessions (controlled by the Media Flow Controller license), and the global resources available in

your system. You can view the global resource pool with **show resource-pool global-pool**. For example, if a server has 5 x 1 Gbps interfaces (1 interface for management and 4 interfaces for media delivery), the maximum bandwidth that can be assigned through various resource pools is 4 Gbps.

The resources allocated to a configured resource pool are removed from the global resource pool (all available resources). If you delete a resource pool, the resources allocated to it are automatically returned to the global resource pool. Whenever a **namespace** is created, it is bound to the global resource pool by default.

NOTE: If you remove an interface from the global resource pool or an interface goes down, the bandwidth resources of the interface are automatically subtracted across all configured resource pools. However, if you add an interface to the global resource pool, you must manually redistribute the additional bandwidth resources.

You cannot manipulate the global resource pool except by deleting or adding the Media Flow Controller license, changing the number of allowed concurrent sessions (**network connection concurrent sessions <integer>**), changing the maximum allowed bandwidth (**network connection max-bandwidth** ***kbps***) or changing the number of configured delivery interfaces (**delivery protocol http interface** ***interface*****...**), or changing disk cache memory limits (by changing disk types).

Once you create a resource pool, these rules apply:

1. Resource pool session configuration:
    a. When a request is received, Media Flow Controller checks for an available resource from the session resource pool. If the maximum number of configured sessions limit for the given namespace is reached, an HTTP “503” (service unavailable) response is returned to this client and the connection is closed.
    b. When a connection is in an idle stage (not transferring any data), it is not counted against session resource pool usage.
    c. Connections established by Media Flow Controller to the origin server(s) are not counted as session resource pool usage.
    d. You can move a namespace between resource pools without needing to deactivate and reactivate the namespace. Likewise, you can change the resource allocation in a resource pool without having to deactivate and reactivate its bound namespace or namespaces. Namespace binding and Resource allocation can be changed on the fly without service interruption.
    e. When you reduce the number of configured resources in the resource pool, and the configured **resource-pool** client session limit is reached, Media Flow Controller does not accept new connections. An HTTP “503” response is returned and the connection is closed; however, all existing sessions on that namespace that are already being handled by Media Flow Controller continue to completion and are not affected. When connections are rejected with the 503 response, you can make the system add the "Retry-After:<N>" header in the response, so the client can

retry after N seconds; this can be configured with **namespace * delivery protocol http connection concurrent session retry-after**.

2. Resource pool bandwidth configuration:

   a. Resource pool represents the aggregated device bandwidth of the Media Flow Controller. It is not limited by the bandwidth capacity of a specific network interface. Media Flow Controller may use one or more interfaces in the device to satisfy the resource requirements of a given namespace. For example, if a Media Flow Controller has 4 x 1 Gbps interfaces and if a namespace is configured with a 2 Gbps bandwidth limit, Media Flow Controller uses 2 interfaces to satisfy the resource requirements.

   b. If a network interface serves one or more resource pools, and it goes down, Media Flow Controller proportionally reduces all resource pools' bandwidth automatically.

   c. You can move a namespace between resource pools without needing to deactivate and reactivate the namespace. Likewise, you can change the resource allocation in a resource pool without having to deactivate and reactivate its bound namespace or namespaces.

   d. Media Flow Controller calculates bandwidth usage once every second. When resource pool bandwidth is over-configured or bandwidth usage surges, Media Flow Controller does not close connections until transactions are complete.

   e. In situations where the resource pool has to utilize bandwidth resources across ports, the ports must be bonded. For example, in a Media Flow Controller with only two 1 Gbps ports, if a resource pool with 1.5 Gbps bandwidth limit is configured, the resource pool will reach the maximum limit at 1 Gbps.

      However, if the two 1 Gbps interfaces in this example are bonded, then the resource pool can reach up to the maximum 1.5 Gbps bandwidth limit.

Bandwidth Management

Configurable bandwidth allocation per namespace allows you to offer specific amounts of bandwidth to tenants. Media Flow Controller calculates the reserved bandwidth for each active session as the Total Bandwidth for a namespace divided by the Total Active Sessions for that namespace (Total Active Sessions = Current Active GET transactions). How much data can be sent out by each active session is calculated every second as part of the accumulate total bytes sent during that second for all sessions.

This bandwidth management is achieved by setting a **resource-pool client bandwidth** value and binding a namespace or namespaces to that **resource-pool**.

Configurable Maximum Sessions

Configurable maximum allowed sessions per namespace restricts the allowable number of incoming client sessions on a per-namespace basis. A default value of 0 (zero) is assumed when no configuration is done. The default value unbinds the number of connections that the namespace handles when no limits are configured for maximum sessions per namespace (that is, by default, the number of connections per namespace is limited by the available resources in the system).

This is achieved by setting a **resource-pool client connection** value and binding a namespace or namespaces to that **resource-pool**.

Resource Monitoring

Media Flow Controller provides key counters on a per-namespace basis that can be queried at frequent intervals by an external agent. It is assumed that an external agent performs a query, stores the values in a database, and can additionally derive other indicators.

Configuring Resource Pools (CLI)

To configure resource pools with the CLI:

1. Configure the resource pool name, maximum allowed bandwidth, and connection limits for a resource pool **client** (a namespace). The numbers of connections configured for a given namespace must not exceed the system-wide configuration for maximum connections (**network connection concurrent session**). A value of **0** (zero) indicates no limit check for the namespace and any number of client connections will be allowed for the namespace. Default for **client bandwidth** is **1Mbps**; for **client connection** it is **10**.

   resource-pool *pool_name* client bandwidth *Mbps*
   resource-pool *pool_name* client connection *integer*

2. Bind a namespace to this resource pool.

   resource-pool *pool_name* bind namespace *namespace_name*

   To unbind a namespace to this resource pool:

   no resource-pool *pool_name* bind namespace *namespace_name*

3. Verify **resource-pool** configurations.

   show resource-pool *pool_name*

4. Verify namespace associations.

   show resource-pool list

5. Save configuration settings.

   write memory

   Example:

```
MFC (config) # resource-pool bronze client bandwidth 1
MFC (config) # resource-pool bronze client connection 10
MFC (config) # resource-pool bronze bind namespace VideoPortal
MFC (config) # show resource-pool bronze
Resource-Pool: bronze
Configured Max Resources
 client sessions: 20
 max-bandwidth (Mbps): 1024
Used Resources
 client sessions: 0
 max-bandwidth (Mbps): 0
MFC (config) # show resource-pool list
Currently defined resource-pool :
      Name       Associated Namespace
=============================================================
```

```
      bronze        VideoPortal
-------------------------------------------------------------
MFC (config) # write memory
```

Related Documentation

- Before You Configure Media Flow Controller on page 20
- Media Flow Controller Delivery Methods on page 9
- Media Flow Controller Multi-Tenancy Management on page 15

Using Boot Disk Mirroring

The Media Flow Controller Release 12.3 supports boot disk mirroring on the VXA2000 Series Content Engine. RAID1 boot disk mirroring, supported only on VXA2000 Series Content Engines, creates an exact copy of the system drive to be able to provide business continuity in the case of a single boot drive failure.

For detailed information about boot disk mirroring, see the *Media Flow Controller 12.3 Installation Guide*.

Related Documentation

- See the *Media Flow Controller 12.3 Installation Guide*

Managing the Media Flow Controller Disk Cache (CLI)

The media caches, or disks, are active and enabled by default and typically require no configuring. Media Flow Controller supports "hot swapping" (system operation is not disrupted); however, you must deactivate and disable disks and caching to change disks. Before you activate or enable a cache, run **show media-cache disk list** to obtain the name assigned to the disk to use in configuration. See **media-cache** in the *Media Flow Controller CLI Command Reference* for CLI details.

NOTE: **When a disk is deactivated, cache latency might increase. Transactions per second (TPS) could drop from 1000 to 700. Perform disk activation and deactivation during a maintenance window when TPS is at its lowest.**

Media Flow Controller supports three cache tiers corresponding to SSD (tier 1), SAS HDD (tier 2), and SATA HDD (tier 3). Hot content generally stays in tier 1 (the highest). Media Flow Controller promotes contents between the cache tiers based on content hotness. As content becomes hotter, it is promoted to the next higher tier. First-time content is always put in the lowest cache tier. The default values are: **tier1** weight = **6**, **tier2** weight = **2**, and **tier3** weight = **1**.

- Analyzing the Disk Cache on page 61
- Disk Cache Problem Solving on page 62
- Disk Cache Error Messages on page 62
- Replacing Bad Disks on page 64

- Correcting Mislabeled Disk Types on page 65
- Inserting New Disks into a VXA Series Media Flow Engine on page 65

Analyzing the Disk Cache

When Media Flow Controller is having caching issues, check the disk cache. To analyze the media disk cache:

1. Get the system-assigned disk names to use in configuration. Use **list** to view all disk drives, their names, physical location, serial number, type, and capacity. Use **cache_name** to view information on the specified cache.

   show media-cache disk (list |*cache_name*)
   show media-cache disk list

   Example:

```
MFC (config) # show media-cache disk list
show media-cache disk list
Device  Type  Tier  Active  Cache  Free Space  State
------  ----  ----  ------  -----  ----------   -----
dc_1   SATA  Tier-3  yes    yes    38890 MiB   cache running
dc_2   SATA  Tier-3  yes    yes    68668 MiB   cache running
Total Free Space: 107486 MiB
MFC (config) # show media-cache disk dc_1
Disk Cache Configuration & Status:
 Device Name/Type: dc_1/SATA
 Cache Tier: Tier-3
 Activated: yesCache Enabled: yes
 Free Space: 2304 MiB
 Disk State : cache running
-------------------------------------
```

2. Disable a disk if you need to pull it for any maintenance purpose; for example, to upgrade to a higher capacity disk, replace a SATA disk with a SAS disk, replace a failed disk, reformat the disk, or if the contents should not be cached.

   media-cache disk *cache_name* cache disable

3. Deactivate a disk cache. Media Flow Controller allows Online Insertion and Removal (OIR) of hard disk drives (HDD) . However, the HDD must be made inactive to be removed. When a new HDD is in the disk, it must be made active and, if applicable, enabled for caching.

   media-cache disk *cache_name* status inactive

4. Put in a new drive and mount it.

   media-cache disk mount-new

5. Find the new (inactive) disk name.

   show media-cache disk list

6. Activate or reactivate a media-cache disk.

   media-cache disk *cache_name* status active

7. Format the disk if it is newly inserted and empty or you do not want to use its contents. Do this after you issue the **mount** command.

media-cache disk *cache_name* format

8. Enable or re-enable a disk for caching.

   media-cache disk *cache_name* cache enable

   Example:

```
MFC (config) # show media-cache disk list
Device    Type   Tier    Active   Cache   Free Space State
------    ----   ----    ------   -----   ---------- -----
dc_1     SATA    Tier-3   yes     yes     38890 MiB cache running
dc_2     SATA    Tier-3   yes     yes     68668 MiB cache running
Total Free Space: 107486 MiB
MFC (config) # media-cache disk dc_2 cache disable
Disk Cache Disabled
MFC (config) # media-cache disk dc_2 status inactive
Disk Deactivated
MFC (config) # media-cache disk mount-new
Message sent to detect for new disks
Please check disk state for status
MFC (config) # show media-cache disk list
Device   Type  Tier   Active   Cache   Free Space   State
------   ----  ----   ------   -----    ---------- -----
dc_1    SATA   Tier-3   yes    yes      38890 MiB cache running
dc_2    SATA   Tier-3   no    no         -        disk has been deactivated
Total Free Space: 107486 MiB
MFC (config) # media-cache disk dc_2 status active
Disk Activated
MFC (config) # media-cache disk dc_2 format
Disk Formatted
MFC (config) # media-cache disk dc_2 cache enable
Disk Cache Enabled
MFC (config) # show media-cache disk list
Device   Type   Tier   Active   Cache   Free Space   State
------   ----   ----   ------   -----   ----------   -----
dc_1     SATA   Tier-3   yes   yes   38890   MiB   cache running
dc_2     SATA   Tier-3   yes   yes   68668   MiB   cache running
Total Free Space: 107486 MiB
```

Disk Cache Problem Solving

When a disk cache error appears, you must first bring the disk down, then bring it back up. To do this, first find cache names, then act on the problem cache:

show media-cache disk list
media-cache disk *cache_name* status inactive
media-cache disk *cache_name* status active
media-cache disk *cache_name* enable

See "Disk Cache Problem Solving" on page 62 for messages you might see at the command line or in a log.

Disk Cache Error Messages

The following are the disk state messages you might see when managing the **media-cache** disk:

Table 6: Disk state messages

Message	Description
DM2_MGMT_STATE_CACHEABLE	"disk cacheable, but cache not enabled"
DM2_MGMT_STATE_INVAL_FORMAT_BEFORE_MOUNT and DM2_MGMT_STATE_FORMAT_UNKNOWN_AFTER_MOUNT	"disk has wrong format hence not cacheable"
DM2_MGMT_STATE_DEACTIVATED	"disk has been deactivated";
DM2_MGMT_STATE_ACTIVATED	"disk has been activated";
DM2_MGMT_STATE_IMPROPER_UNMOUNT and DM2_MGMT_STATE_IMPROPER_MOUNT	"soft disk error, try to clear"
DM2_MGMT_STATE_CACHE_RUNNING	"cache running";
DM2_MUST_FORMAT	" Disk Cache Enable Failed - Disk Cache must be formatted before enabling"
DEFAULT	"unknown state, please try again a little later";

The following messages might display in a log:

Table 7: Log messages

Message	Description
NKN_DM2_DISK_ADMIN_ACTION	A previous command caused the disk to be in the "cache enabled" state.
NKN_DM2_EMPTY_CACHE_TIER	A request was sent to a disk tier with no enabled disks.
NKN_DM2_EVICTION_SKIPPED	An eviction request was made but the request had some type of error in it.
NKN_DM2_WRONG_CACHE_VERSION	Attempted to enable a cache with the wrong version number. An older version was not converted to the current version. Media Flow Controller does not downgrade cache versions.
NKN_DM2_DISK_CACHE_NOT_FOUND	An unknown disk cache name was given to the CLI.
NKN_DM2_INVALID_MGMT_REQUEST	The command attempted was not valid for the given state of the disk. This is a generic error.
NKN_DM2_DISK_DEVICE_NOT_FOUND	During a status active command, the system could not find the requested device.

Table 7: Log messages *(continued)*

Message	Description
NKN_DM2_CONVERT_FAILED	When Media Flow Controller is starting or a drive is made " status active," format conversion might have occurred. Media Flow Controller returns this error if the conversion failed.
NKN_DM2_MUST_CACHE_DISABLE	A drive is in the "cache enabled" state and a status inactive command was given.
NKN_DM2_MUST_CLEAR	A drive is in an error state of any kind and a cache enable command was given.
NKN_DM2_MUST_STATUS_ACTIVE	If a drive is in the status inactive state and a cache enable command was given.

Replacing Bad Disks

To replace a bad disk, first disable and inactivate the disk, and then add a new disk. This procedure uses **dc_bad** for the disk name; replace the disk name appropriately.

To replace a bad disk:

1. Discover the name of the bad disk.

   show media disk list

2. Disable the bad disk by name.

   media-cache disk dc_bad cache disable

3. Make the bad disk inactive.

   media-cache disk dc_bad status inactive

4. Pull the drive, put in a new drive, and mount the new disk.

   media-cache disk mount-new

5. Find the name of the inactive disk.

   show media-cache disk list

6. Activate the new disk.

   media-cache disk dc_new status active

7. Format the new disk; this takes approximately 5 minutes.

   media-cache disk dc_new format

8. Enable the new disk.

   media-cache disk dc_new cache enable

CAUTION: This procedure will not work when adding one or more disks to a Juniper Networks VXA Series in a slot that is not configured. In this case,

follow the instructions in the VXA Series hardware guides for adding a new disk using the Adaptec Utility, or see "Inserting New Disks into a VXA Series Media Flow Engine" on page 65.

Correcting Mislabeled Disk Types

Occasionally, SSD disk types may be mistakenly labeled as SATA disk types. If this happens, you can correct the problem by relabeling the disks.

To correct the disk type to SSDs for those disks mistakenly labeled as SATA:

1. Run the following command and note the label for the SATA disks; for example **dc_n**:

   show media-cache disk list

2. Disable each mislabeled disks.

   media-cache disk *dc_n* **cache disable**

3. Configure the correct disk type for each mislabeled disk.

   media-cache disk *dc_n* **disk-type SSD**

4. Restart the delivery service.

   service restart mod-delivery

5. Wait for 1 minute. Then, format each relabeled disk.

   media-cache disk *dc_n* **format**

6. Enable each relabeled disk.

   media-cache disk *dc_n* **cache enable**

7. Save your work.

   write memory

Inserting New Disks into a VXA Series Media Flow Engine

The task "Replacing Bad Disks" on page 64 does not work for VXA Series hardware when adding one or more new disks to a slot that was never used (configured) because Media Flow Controller only searches for new disks at manufacture time. Use this procedure instead.

To add a disk, or disks, on a VXA Series Media Flow Engine to an unconfigured slot:

1. From Enable or Configuration mode in the CLI, check the current drive list. You may want to note the drives you have installed.

   show media-cache disk list

2. Insert the new disks and reboot.

   reload

3. When prompted to open the Adaptec Utility, press Ctrl+A to open the utility. The **Adaptec Utility** menu appears.

4. Select **Array Configuration Utility**.

   The **Configuration Change** confirmation window appears.

5. Select **Accept**.

   The **Configuration Utility Main** menu appears.

6. Use the arrow keys to select **Initialize Drives**, and press Enter.

   A list of discovered drives appears.

7. Select the drive or drives that you added, and press Enter.

   A warning message appears.

   If you are unsure which drives to select, use the Esc key to exit to the **Adaptec Utility** menu and use the **Disk Utilities** function to identify the drives.

8. Enter **Yes**.

   The **Initializing is Done** message appears.

9. Press any key to continue.

   The Configuration Utility Main menu appears.

10. Select **Create JBOD**, and press Enter.

    A list of drives appears.

11. Select the new drives, and press Enter.

    A confirmation window appears.

12. Enter **Yes**.

    The **Adaptec Utility** menu appears.

13. Press Esc to exit the Adaptec Utility.

    The system reboots automatically.

14. Using the Media Flow Controller CLI, mount the new disks and activate them.

    ```
    media-cache disk mount-new
    show media-cache disk list
    media-cache disk disk_name activate
    ```

15. Verify that the new disks are properly mounted.

    ```
    show media-cache disk list
    ```

NOTE: The documented disk replacement procedure for replacing a bad disk on VXA Series works correctly provided that Media Flow Controller can still identify the disk (**show media-cache disk list**).

Related Documentation

- Before You Configure Media Flow Controller on page 20
- Media Flow Controller Minimum System Requirements on page 4

Administering Media Flow Controller Overview (CLI)

You can perform standard Media Flow Controller administrative tasks using the CLI:

Checking Media Flow Controller Version and Status

You can check the release version and system status and other conditions with the show version command. For example:

```
MFC (config) # show version
Copyright (c) 2008-2010 by Juniper Networks, Inc
Product name:   mfc
Product release:  mfc-12.1.0-dv
Build ID:     302_12730_216
Build date:    2010-10-19 10:31:36
Target arch:   x86_64
Built by:     build@build03
Uptime:     6m 11s
Product model:   standard
Host ID:     D0F4D91
System memory:   921 MB used / 1088 MB free / 2009 MB total
Swap:      380 MB used / 648 MB free / 1028 MB total
Number of CPUs:  2
CPU load averages: 1.86 / 1.33 / 0.59
```

Saving and Applying Configurations, and Resetting Factory Defaults (CLI)

You can save a binary file with all current configuration data, and later use it to restore the system configuration. You can also reset custom configurations to their factory defaults, upload a saved configuration, and import a configuration from another Media Flow Controller.

To save and apply a configuration using the CLI:

1. Save a configuration to a file; use the **no-switch** option to leave the current configuration active. Use the **show configuration files** command to verify the saved filename.

   configuration write to *file_name* no-switch

2. Use SCP to send the just-saved configuration file to a server (must have SCP installed).

   configuration upload *file_name URL*

3. Use SCP to send the supporting configuration files (that is, service files) to a server.

configuration files upload scp://*username*[:*password*]@*hostname/path/filename*

4. When you are ready, fetch the service files.

   configuration service files fetch *download URL*

5. Fetch the saved configuration file.

   configuration fetch *URL/file_name*

6. Verify that you have the saved configuration file.

   show configuration files

7. Switch to the saved configuration.

   configuration switch-to *file_name*

   Example:

```
MFC (config) # configuration write to 04_01_09 no-switch
MFC (config) # show configuration files
04_01_09
initial (active)
initial.bak
MFC (config) # configuration upload 04_01_09 scp://joe@example.com/home/joe/
Password: *******
MFC (config) # configuration delete 04_01_09
MFC (config) # show configuration files
initial (active)
initial.bak
MFC (config) # configuration fetch scp://joe@example.com/home/joe/04_01_09
Password: *******
MFC (config) # show configuration files
04_01_09
initial (active)
initial.bak
MFC (config) # configuration switch-to 04_01_09
MFC (config) # show configuration files
04_01_09 (active)
initial
initial.bak
```

8. Merge the common settings from a given configuration file to the active configuration file.

   configuration merge *file_name*

9. Revert the active configuration to either the factory defaults or the last saved configuration.

   Use the **keep-basic** option to preserve licenses and SSH host keys, use the **keep-connect** option to preserve whatever is necessary to maintain network connectivity to the system: interfaces, routes, and ARP. You can use either or both options.

   configuration revert (factory | saved) [keep-basic] [keep-connect]

Rebooting Media Flow Controller (CLI)

You can either reboot or shut down Media Flow Controller. If you use the **shutdown** command, the system does not reboot until it is power cycled.

To reboot Media Flow Controller using the CLI:

1. Reboot or shut down the system.

   reload

2. Set boot parameters.

   Optionally specify a default location from which the image boots; since there are only two locations to choose from, the options are **1** and **2** for location ID. If you use the **next** option, set the boot location to be the next one after the one currently booted from.

   boot (location *location_ID* **| next)**

3. View boot parameters.

   boot ?

4. View current settings.

   show boot

Upgrading Media Flow Controller (CLI)

When upgrades are available, Juniper Networks will broadcast the upgrade URL to use in this procedure. Upgrade preserves the current, saved, configurations; however, you might still want to save the current configuration to a file on another system by following the previous procedure.

NOTE: If disk partitions are changing, a remanufacture or install is required; see "Saving and Applying Configurations, and Resetting Factory Defaults (CLI)" on page 67 along with the appropriate installation instructions described in the *Media Flow Controller Installation Guide*.

To upgrade Media Flow Controller using the CLI:

1. Get the upgrade image file with the supplied URL.

   image fetch *URL/filename*

2. Install the image.

   1. Enter enable CLI mode.

      > **enable**

   2. Enter configure terminal CLI mode.

      # **configure terminal**

3. Enter the **image install** command.

   #**image install** ***filename***

3. Verify which boot image contains the upgrade.

   # **show images**

4. Switch to the boot partition containing the upgrade image, if needed.

   # **image boot next**

5. Reboot to that partition.

   # **reload**

6. Verify that the new image is booted.

   # **show version**

Manually Implementing Default Value Changes in 2.0.x Releases

The defaults for several commands changed in 2.0.x Releases. The new defaults are not applied to an upgrade when you save the configuration settings. You can use these commands to implement new defaults in Media Flow Controller 2.0.x Releases. Change variables as needed (such as **namespace** ***name***), and ensure that you do not overwrite a deliberate (non-default) configuration.

To manually implement 2.0.x Release default value changes, run these commands from the CLI, in **Configure** mode:

```
accesslog format %c %h %V %u %t "%r" %s %b %N "%{Cache-Control}i" "%{Pragma}i"
  "%{Cache-Control}o" "%{Pragma}o" "%{Vary}o" %y
delivery protocol allow-req all
delivery protocol http conn-pool origin max-conn 4096
delivery protocol http conn-pool origin timeout 90
delivery protocol http connection persistence num-requests 32
namespace name delivery protocol http origin-fetch cache-age-default 0
namespace namedelivery protocol http origin-fetch cache-fill
client-driven
namespace name delivery protocol http origin-fetch content-store
media object-size 4096
namespace name delivery protocol http origin-request host-header
inherit incoming-req deny
network concurrent session 64000
ram-cache revalidate-minsize 1
```

Configuring the Web Interface (CLI)

Before you configure the Media Flow Controller Web interface, see "Before You Configure Media Flow Controller" on page 20.

The **web** command lets you specify timeouts, ports, and protocols for access to the Media Flow Controller Web interface, also referred to as the Management Console. See **web** in the *Media Flow Controller CLI Command Reference* for CLI details. We highly recommend setting the Web interface port to something other than 80 as that port is used for service traffic; port 8080 is the default, and recommended for Web interface access.

To configure the Web interface:

1. The Web interface (Management Console) is **enabled** by default. If you need to, you can disable it or re-enable it.

   web no enable
   web enable

2. Set the automatic logout when the Web interface is idle. The default is **15** minutes.

   web auto-logout *number_of_minutes*

3. Enable HTTP, HTTPD, and HTTPS and set a port or listen interface (HTTPD only); all are **enabled** by default. The default HTTP port is **8080**, the default HTTPD listen interface is **eth0** (used for management), and the default HTTPS port is **443**.

   web http enable
   web http port *port_number*
   web httpd listen enable
   web httpd listen interface *interface_name*
   web https enable
   web https port *port_number*

4. Configure Web session cookie options.

   The **renewal** option is the length of time before Web session cookies are automatically regenerated, The default is **30 minutes**. The **timeout** is the time after which a session expires. The default is **900 seconds** or **15 minutes**.

   web session renewal *number_of_minutes*
   web session timeout *number_of_minutes*

   Example:

   ```
   MFC (config) # web auto-logout 9000
   MFC (config) # web http port 8080
   MFC (config) # web httpd listen interface eth0
   MFC (config) # web https port 443
   MFC (config) # web session renewal 60
   MFC (config) # web session timeout 9000
   MFC (config) #
   ```

Configuring the Web Interface Proxy (CLI)

Configure Web interface proxy settings if your Web server uses a proxy. The **web proxy** command lets you specify authentication for access to the Media Flow Controller Web interface, and a host for the proxy.

The Web **proxy auth** options do not take effect without a configured Web proxy host. See **web** in the *Media Flow Controller CLI Command Reference* for CLI details.

To configure the Web interface proxy:

1. Set the Web proxy authentication options; **basic** is the HTTP basic authentication. The default is **none**. If you set the **authtype** to **basic**, then use **auth basic** to set the **password** and **username** to be authenticated.

   web proxy auth authtype (none | basic)

```
web proxy auth basic password plaintext_password
web proxy auth basic username username
```

2. Set the Web proxy **host** and, optionally, the **port** (the default is 1080). Setting the Web proxy host enables the Web proxy. If set, this proxy accepts HTTP and FTP downloads (the FTP default port is 21).

   ```
   web proxy host IP_address [port TCP_port]
   ```

   Example:

   ```
   MFC (config) # web proxy auth authtype basic
   MFC (config) # web proxy auth basic password 123
   MFC (config) # web proxy auth basic username admin
   MFC (config) # web proxy host 123.45.10.9 port 8080
   MFC (config) #
   ```

Related Documentation

- Before You Configure Media Flow Controller on page 20

Configuring Caching All Contents for a Website (CLI)

Configure Media Flow Controller to cache all contents of the website *www.example.com*. Change variables, such as *domain* and *origin-server*, as needed. Before you configure Media Flow Controller to cache all contents of a website, see"Before You Configure Media Flow Controller" on page 20 and "Media Flow Controller System Configuration Overview (CLI)" on page 28. This example does not include Media Flow Controller system setup.

To configure caching of all contents for a website:

1. Set up DNS records or Hosts entries so that incoming requests are routed to Media Flow Controller. Be sure to set the Media Flow Controller DNS server so it is not the same as your Web server's DNS server (to prevent looping).
2. Configure a **namespace** with **domain www.example.com** and **match uri-prefix /** (slash), and an **origin-server**; make the namespace **active**.

   ```
   MFC (config) # namespace example
   MFC (config namespace example) # domain www.example.com
   MFC (config namespace example) # match uri /
   MFC (config namespace example) # origin-server http www.exampleOS.com
   MFC (config namespace example) # status active
   MFC (config namespace example) # exit
   ```

   The **match uri-prefix** of just a **/** (slash) tells Media Flow Controller to use these namespace rules for all incoming requests to *domain www.example.com*, since all incoming requests have a **/** (slash) in them.

TIP: If unsure what port your **origin-server** is using, use standard Linux shell commands (for example, **netstat –nl**) to determine the port, and then configure it along with the **origin-server**, if it is not the default. If you need to change the **origin-server**, or any **namespace** setting, simply enter the new setting.

There are many **namespace** options, including **cache-inherit**, **delivery protocol**, **origin-fetch**, and so forth. See "Creating a Namespace and Setting Namespace Options (CLI)" on page 76 for more information, and **namespace** in the *Media Flow Controller CLI Command Reference* for CLI details.

3. Repeat the configuration twice, if needed, to cover requests coming in to " example.com" (without the " www" prefix) and the IP address for example.com (if requests can come in that way).

4. Verify the configuration.

```
show namespace example

Namespace: example
  Active: yes
  Precedence: 0
  Domain Name: www.example.com
  Proxy Mode: reverse
  Match Type:
    URI-Prefix - /
  Origin-Server: http://www.exampleOS.com:80
  Delivery Protocol: HTTP Status Enabled: yes
  Origin Fetch Configuration:
    Cache-Age (Default): 28800 (seconds)
    Cache Age Threshold: 60 (seconds)
    Cache-Directive: follow
    Object Size Threshold: NONE (Always Cached)
    Modify Date Header: deny
  Origin Request Configuration:
    Cache-Revalidate: permit
      Force end-to-end cache revalidation : Disabled
      Use 'Date' Header when Last-Modified is not present: no
    Convert HEAD to GET: permit
    Host-header Inherit: deny
    Set X-Forwarded-For Header : yes
  Client-Request Configuration: {
    Allow objects with a query-string to be cached: no
  Client-Response Configuration :
  Delivery Protocol: RTSP Status Enabled: no
  Origin Fetch Configuration:
    Cache-Age (Default): 28800 (seconds)
    Cache Age Threshold: 60 (seconds)
    Cache-Directive: follow
    Object Size Threshold: NONE (Always Cached)
  Virtual Player:
```

5. From the browser, initiate a connection to Media Flow Controller for *www.example.com*. Check the access log to see that Media Flow Controller processed it. You can view the error log and the access log using the Media Flow Controller Web interface; use a browser to open the Media Flow Controller IP address on port 8080 and login (the default log in is **admin** / no password) and open the **Service Logs** tab.

Related Documentation

- Before You Configure Media Flow Controller on page 20
- Media Flow Controller System Configuration Overview (CLI) on page 28

CHAPTER 3

Configuring Namespaces (CLI)

Creating a Namespace and Setting Namespace Options (CLI)

Create namespaces to define fine-grained delivery policies, including optionally adding a custom virtual player. You must create a namespace for each origin server and delivery criteria scheme you use. You can create up to 256 namespaces in one Media Flow Controller (this includes the internal namespace mfc_probe). See **namespace** in the *Media Flow Controller CLI Command Reference* for CLI details. Before you configure Media Flow Controller namespaces, see "Before You Configure Media Flow Controller" on page 20.

If your namespace will use a server map, see "Configuring Server Maps (CLI)" on page 232 for information about creating server maps.

If your namespace will use a resource pool, see "Creating Resource Pools (CLI)" on page 56.

NOTE: **Avoid ambiguous namespace name. Avoid using an URI that matches more than one hit.**

Related Documentation

- Media Flow Controller Server Map Overview on page 223
- Example: Configuring Media Flow Controller Namespaces (CLI) on page 111

Namespace Selection Logic Overview

This section describes the namespace selection logic for the Media Flow Controller.

When you create a namespace, you define the domain (**any** is the default) and define a match criteria. The namespace match criteria can be any one of the following criteria; you cannot define a combination of namespace match criteria:

- exact URI match
- URI regex
- header name
- header name regex
- query string

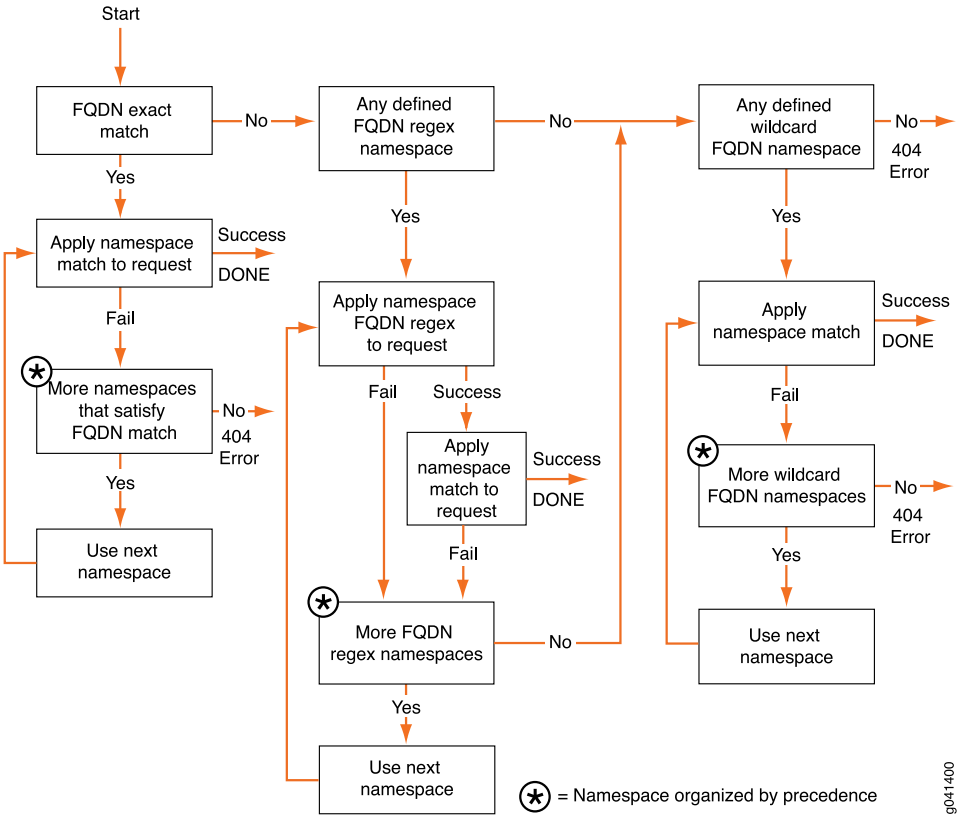
- query string regex
- virtual host

See *Chapter 3: Configuring Namespaces* for information on creating namespaces.

The Media Flow Controller then uses the defined domain and match criteria to select the specific namespace.

The selection logic is shown in Figure 6 on page 77.

Figure 6: Namespace Selection Logic



The top box in each column is applying the domain selection logic on the request, followed by application of namespace match criteria. When the Media Flow Controller successfully applies a namespace match, the process is done. If the device fails to find any namespace match, you see the following message: **404 Error**.

Related Documentation

- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Using namespace match uri regex on page 97
- Using namespace match virtual-host on page 98

Using namespace bind policy

Bind a configured, and committed, **policy** to this namespace; only committed polices can be bound. See **policy** in the *Media Flow Controller CLI Command Reference* for details. You can bind or unbind policies from a namespace without first changing the status to inactive. The policies bound to the namespace are executed last. Examples, from namespace prefix mode:

- Bind a policy to the namespace:

  namespace *name* bind policy *policy_name*

- Unbind a policy to the namespace:

  no namespace *name* bind policy *policy_name*

Related Documentation

- Media Flow Controller Server Map Overview on page 223
- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

Using namespace Cache Pinning

Use the namespace pinned-object option to preload and pin an object in the Media Flow Controller cache to improve cache performance and handle flash crowds.

To pin an object:

- Enable pinning under a namespace
- Configure a custom HTTP head name, which Media Flow Controller uses to differentiate the objects to be pinned from other objects
- Configure origin server to insert the custom HTTP header in the response for objects that need to be pinned
- Configure preload to preload the object into the Media Flow Controller cache

To configure cache pinning:

- Specify the maximum namespace cache size (in GB) allocated for storing pinned objects:

  namespace *name* pinned-object cache-capacity *xyz*

- Specify the maximum allowed size (in KB) for a pinned object in the namespace:

  namespace *name* pinned-object max-obj-size *xyz*

- Enable or disable object pinning for the namespace:

  namespace *name* pinned-object status *enable*

- Specify the custom header name for cache pinning (the custom header has to be inserted by the origin server to indicate that Media Flow Controller has to "pin' the object in the cache file system):

  namespace *name* pinned-object pin-header *name*

- Specify a header name for object validity start time (not tied to pinning functionality):

  namespace *name* object validity-begin-header *header-name*

Related Documentation

- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

Using namespace cache-inherit

The **namespace cache-inherit** option recovers data from a deleted namespace under the following circumstances:

- You delete an existing namespace, rename it, and want to use the data cached under it.
- You delete a namespace by accident, want to re-create it, but do not want it to have a new UUID. In this case, you delete the namespace and its associated UUID, and force the UUID of an existing namespace for the one you are creating. To do this:

1. Enter **show namespace list** to gather the list of “Currently defined” and “Deleted/non-existing” (but whose cache content still exists) namespaces and their associated UUIDs.
2. Enter **namespace** *name* **cache-inherit** ***existing or non-existing namespace integer***.

For example: **show namespace list** (namespace **test2** inherited namespace **test** cache):

show namespace list

```
Currently defined namespaces :
  Name : UID
  new_ns : /new_ns:b3ad64a8 (inactive)
  test2 : /test:1250bcac (active)
  testIE : /testIE:b911b24 (inactive)
  ------------------------
List of unmapped/deleted namespace UIDs (if any)
 non-exsiting1: /test3:954ef8aa
```

Related Documentation

- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

Using Namespace to Handle Cookies

Media Flow Controller allows you to cache or not cache request URIs with cookies, and to specify responses to have a cookie set that causes the response to be cached.

You define when an incoming object with a cookie can be cached using the **namespace *name* delivery protocol http client-request cookie-action** options (**cache** and **no-cache**). Setting this option to **no-cache** means "tunnel the response."

You can configure Media Flow Controller to set a cookie on certain headers to cache or not cache objects when media is delivered from Media Flow Controller using the **namespace *name* delivery protocol http origin-fetch header set-cookie** option.

Other unknown strings including "Cache-Control: public," "Cache-Control: no-cache=set-cookie", and so on, Media Flow Controller ignores these and forwards them as-is to the client.

Related Documentation

- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

Controlling Cookie Cache Behavior (CLI)

Media Flow Controller can be configured to act on cookies in the (incoming) request path or the (outgoing) response path.

- About HTTP State Management and Cookies on page 80
- Controlling Cookies In the Request Path on page 81
- Controlling Cookies In the Response Path on page 81

About HTTP State Management and Cookies

HTTP servers respond to each client request without associating that request to previous or subsequent requests. The state management mechanism allows clients and servers to exchange state information by placing HTTP requests and responses within a larger context termed as a session. HTTP servers provide each user session with a " cookie" to track the current state of the session. To better understand HTTP cookies, refer to RFC 2965. **Cookie2Set-Cookie** or **Set-Cookie2** are headers always seen in responses from origin towards clients. Web applications use these headers to exchange state information. Clients MUST forward the cookie value in subsequent requests to the origin so the origin applications can track the state of individual clients. Origin servers also optionally set the **Cache-Control** header in responses so that caching proxies, or the end-user client, may or may not cache the response objects as configured.

Cookie or **Cookie2** - are headers always seen in requests from client towards origin.

NOTE: **By default, Media Flow Controller follows the instructions of the Cache-Control header in the HTTP request or the HTTP response. However, these can be overridden by using the CLI or the policy engine as outlined below.**

Media Flow Controller provides configurable options using the command-line interface (CLI) and the Policy engine to control caching for objects with cookies. Filtering behavior control is provided by the Policy Engine interface since it allows for building a rich set of

rules using a programmable interface; see "Policy Engine Overview" on page 245, for more information.

Controlling Cookies In the Request Path

On the request path, you can configure Media Flow Controller to cache or not cache an associated response containing a cookie:

```
namespace name
  delivery protocol http
    client-request cookie action cache
```

With this CLI command, Media Flow Controller caches responses when clients send a **Cookie** or **Cookie2** header in the request. For objects that are already cached, no object is revalidated as long as the object is still " fresh" (within configured limits). Moreover, if the cached response has a **Set-Cookie** or **Set-Cookie2** header, the response to the client will contain the same headers.

Use the following to cache objects that have cookies in the response:

```
namespace name
  delivery protocol http
    client-request cookie action no-cache
```

Use this CLI command to not cache any objects that are likely to have cookies in the request.

Controlling Cookies In the Response Path

On the response path, Media Flow Controller can be configured to control caching of responses based on the **Set-Cookie** or **Set-Cookie2** headers received in the response from the origin server.

```
namespace name
  delivery protocol http
    origin-fetch header set-cookie any cache deny
```

This CLI command configures Media Flow Controller to bypass caching any responses received from the origin if the response contains a **Set-Cookie** or **Set-Cookie2** header.

```
namespace name
  delivery protocol http
    origin-fetch header set-cookie any cache permit
```

This CLI command configures Media Flow Controller to force caching of responses received from the origin even if the responses contain a **Set-Cookie** or **Set-Cookie2** header.

CAUTION: **Use of this option causes Media Flow Controller to hand-out the same cookie to multiple users.**

When the caching policy is set to allow caching of objects with **Set-Cookie** or **Set-Cookie2** headers, Media Flow Controller always attempts to follow the Cache-Control directive.

Using namespace delivery protocol http client-request cache-hit action revalidate-async

You can configure the Media Flow Controller to serve expired objects to the client. This feature allows the cache to act as an origin server, thus reducing the origin server dependency.

This feature is typically used in reverse-proxy deployments and is configurable at a per namespace level.

When this feature is enabled and if the cache contains expired or stale objects, the clients are served with that expired or stale object. Simultaneously, the system triggers an offline, or asynchronous, revalidation request for that expired or stale object. During the asynchronous revalidation, one of the following conditions occurs:

- When the origin server responds that the object is not modified, the cached objects are updated with the incoming expiry time from the origin server and sent to subsequent clients.
- When the origin server responds that the object is modified, the cached object is invalidated. At the next client request, the object is fetched from the origin server, and the system caches the fresh object. The fresh object is delivered to the client.
- When the origin server does not respond (for example, for those connect/read timeout cases) or responds with a 5xx code, the cached object is not invalidated. Those unmodified or expired objects are served to subsequent client requests.
- When the origin server responds with a 404 (file not found) response, the cached object is invalidated and the same error is forwarded to the clients.

NOTE: if the **namespace** *namespace* **object revalidate all** command is enabled, all cached objects are considered expired. Enabling this revalidate-async feature overrides the revalidation-all feature.

Note the following interactions this feature takes with related features in the software:

- Because this is a feature used in reverse proxy deployments, the revalidation-always feature should *not* be enabled simultaneously with this revalidate-async feature. That is, when you enable the revalidate-async feature, ensure that you have not configured the **namespace** *namespace* **delivery protocol http client-request cache-hit action revalidate-always** command in the same namespace.
- When you configure the connection timeout or read timeout to a number as high as 60 seconds, the revalidation response might be delayed. In this case, subsequent clients are also served the expired contents from the cache.
- When this feature is enabled, the cache acts as the origin server. Thus, parameters for the origin server, such as cache-control: max-age, are ignored.

- With this feature enabled, the software tunnels requests that a client sends with a cache-control: no-cache header.
- During revalidation, if the origin server responds that the object is modified or there is a permanent failure (such as a 404 error), the active client download might be terminated. (The cached copy is modified or removed, which causes the active download to terminate.)

To configure the revalidate-async feature, enter the following command:

- **[no] namespace** *namespace* **delivery protocol http client-request cache-hit action revalidate-async**

You can view the number of objects asynchronously revalidated by entering the **show** *namespace* **counters** command. Additionally, the access log contains a field that shows when the user is served with one of these objects.

Related Documentation

-

Caching Client Requests Using Tunnel Override

By default, the system does not cache a client request sent with an Authentication, Cache-Control, or a Cookie header. See *Factors Influencing the Cache-ability of an Object*.

This bypass mechanism is called "tunneling." For transparent proxy deployments, the main benefit of cache tunneling is that by overriding certain tunneled transactions, the cache hit ratio is improved for the remaining transactions. All of the client request tunnel-override options are disabled by default.

To enable the client request tunnel-override options, use the **namespace** ***name*** **delivery protocol http client-request tunnel-override** (**auth header** | **cache-control** | **cookie**) CLI commands. See also the "(namespace) delivery protocol http client-request" section in the *Media Flow Controller CLI Command Reference*.

1. Caching Authorization Header Client Requests on page 83
2. Caching Cache Control Client Requests on page 84
3. Caching Cookie Client Requests on page 85

Caching Authorization Header Client Requests

The HTTP Authorization header allows the client to authenticate itself with the server after receiving a 401 response. By default, caching a client request when an Authorization header is disabled.

To enable client request Authorization-header caching:

1. Enter the CLI configuration mode.

   > **enable**

   # **config terminal**

2. Enable the auth-header tunnel-override option.

   (config) # **namespace *name* delivery protocol http client-request tunnel-override auth-header**

   The system caches client requests containing the Authorization header—Tunnel_14.

3. View the tunnel-override auth-header configuration.

   (config) # **show namespace *name***

```
...
 Client-Request Configuration:
   Secure Delivery(HTTPS over SSL):Disable
   Plain-text Delivery:Enable
   Allow objects with a query-string to be cached: no
   Cache control max age : 0
   Cache control max age action : serve-from-origin
   Allow objects with a cookie header to be cached: yes
   Revalidate object always : No
   Tunnel All Requests : Disabled
   Action on Connect method :  TUNNEL
   Tunnel-Override Configuration:
   AUTH Header
...
```

Caching Cache Control Client Requests

The Cache-Control header specifies directives that must be obeyed by all the caching mechanisms along the request/response chain. The Pragma general header field includes directives that might apply to any recipient along the request/response chain. The Pragma: no-cache request provides backward compatibility with an HTTP 1.0 server. Clients should include both headers when a server is not known to be HTTP 1.1 compliant.

By default, caching client requests with Cache-Control headers is disabled.

To enable client request Cache-Control header caching:

1. Enter the CLI configuration mode.

   > **enable**

   # **config terminal**

2. Enable the tunnel-override cache-control option.

   (config) # **namespace *name* delivery protocol http client-request tunnel-override cache-control**

   The system caches requests containing either Cache-Control: no-cache or Pragma: no-cache Tunnel_17 and Tunnel_18.

3. View the tunnel-override cache-control configuration.

   (config) # **show namespace *name***

```
...
 Client-Request Configuration:
   Secure Delivery(HTTPS over SSL):Disable
```

```
  Plain-text Delivery:Enable
  Allow objects with a query-string to be cached: no
  Cache control max age : 0
  Cache control max age action : serve-from-origin
  Allow objects with a cookie header to be cached: yes
  Revalidate object always : No
  Tunnel All Requests : Disabled
  Action on Connect method :  TUNNEL
  Tunnel-Override Configuration:
  Cache-control
...
```

Caching Cookie Client Requests

The cookie header captures data about the state of the client web browser. By default caching cookie client requests is disabled.

To enable caching cookie client requests:

1. Enter the CLI configuration mode.

   > **enable**

   # **config terminal**

2. Enable the cookie option.

   (config) # **namespace *name* delivery protocol http client-request tunnel-override cookie**

   The system caches client requests containing the cookie header Tunnel_15.

3. View the tunnel-override cache-control configuration.

   (config) # **show namespace *name***

```
...
 Client-Request Configuration:
   Secure Delivery(HTTPS over SSL):Disable
   Plain-text Delivery:Enable
   Allow objects with a query-string to be cached: no
   Cache control max age : 0
   Cache control max age action : serve-from-origin
   Allow objects with a cookie header to be cached: yes
   Revalidate object always : No
   Tunnel All Requests : Disabled
   Action on Connect method :  TUNNEL
   Tunnel-Override Configuration:
   COOKIE Header
...
```

Related Documentation

- Caching Origin Fetch Requests Using Tunnel Override on page 93
- Factors Influencing Whether an Object Will Be Cached on page 13

Using namespace delivery protocol (http|rtsp) origin-fetch cache-age

The **namespace** ***name*** **delivery protocol http origin-fetch cache-age** argument allows you to set granular cache aging policies based on content type, as well as a default cache age. The **cache-age** for each **content-type** must be specified separately with positive integers. See **namespace** in the *Media Flow Controller CLI Command Reference* for details. For example:

- **cache-age content-type-any 28800**

  Irrespective of content type, override your configured **cache-age-default** or, if **cache-age-default** is unconfigured, set the Max-Age for any content type to **28800** seconds. If the content request does not specify a Max-Age, set it to Max-Age **28800**.

- **cache-age content-type application/flv 2880**
  cache-age-default 900

  When **content-type** is **application/flv**, set Max-Age to **2880** seconds. For all other content-types use **default** configuration (**900** seconds in the example). If the received Max-Age is set, use that value.

- **cache-age content-type application/flv 28800**
  cache-age content-type application/mov 2880
  cache-age content-type application/3gp 288
  cache-age content-type application/f4v 28
  cache-age-default 900

  When **content-type** is **application/flv**, set Max-Age to **28800** seconds; for **application/mov**, set Max-Age to **2880** seconds; for **application/3gp**, set Max-Age to **288**; and for **application/f4v**, set Max-Age to **28**. For all other content-types, use **default** configuration (**900** seconds in the example). If the received Max-Age is set, use that value.

- **cache-age content-type application/qmx 60**
  cache-age content-type application/qss 288000

  When **content-type** is **application/qmx**, set **cache-age** to **60** seconds; for **application/qss**, set **cache-age** to **288000** seconds. For all other content-types, use **default** configuration: if Max-Age is not set in the data coming from origin, set it to the configured **default** value (**28800** if unspecified). If the received Max-Age is set, use that value.

The Media Flow Controller first checks for a specified match for the **content-type**. If there is no specific match, the systems uses the value configured with the **content-type-any** command for the maximum age.

You might see a case where the administrator sets the age of objects using the **cache-age content-type** ***string*** command to a certain value and the Cache-Control:max-age header in the origin's response is a different value. In this case, the system uses the value configured in the **cache-age content-type** ***string*** command, and not the value in the Cache-Control:max-age header. Also, the system sends the value configured in this command in the response to the user.

You might configure a value for the **cache-age content-type-any** command as well as a separate value for the **cache-age content-type *string*** command. If the origin response content type does not match that configured with the ***string*** value, the system overwrites the original value with that configured using the **cache-age content-type-any** command.

Related Documentation

- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

Using namespace delivery protocol http origin-fetch-cache-control

The **namespace** *name* delivery protocol http origin-fetch cache-age argument allows you to override standard Cache-Control headers. The "X-Accel-Cache-Control" header, which supplies a value for the duration of caching for that object, can be used to do this. See **namespace** in the *Media Flow Controller CLI Command Reference* for details.

When processing HTTP 304 (Not Modified) responses from origin, the response headers are processed in the following order:

1. If "X-Accel-Cache-Control" header is present in the 304 response, and if the namespace is configured to use that header for cache control, then Media Flow Controller uses the "X-Accel-Cache-Control" header value.
2. If the response does not have the "X-Accel-Cache-Control" header, Media Flow Controller works based on the available " Cache-Control" header directive.
3. If the response does not have either the "X-Accel-Cache-Control" or the "Cache-Control" headers, Media Flow Controller defaults to policies configured for that namespace.

Related Documentation

- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

Using namespace delivery protocol http origin-fetch cache-fill

When the origin server is known to not support byte-range requests, do not configure the namespace **origin-fetch cache-fill** policy to be **client-driven**. Instead, use the **cache-fill aggressive** option.

If Media Flow Controller is configured with a namespace **origin-fetch cache-fill client-driven** policy, and a partial file is present in the cache, when the client makes a GET request for the same object, Media Flow Controller delivers to the client the partial file available from cache and makes a byte-range request to the origin server for the remaining bytes. If the origin server does not support byte-range requests, it responds with "200 OK" for the byte-range request, the partial file is not deleted from the cache, and the media delivery is stopped.

Related Documentation

- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

Using namespace delivery protocol http origin-fetch Cache Responses

Using this CLI command, you can cache the non-2xx responses from the origin servers, which helps you to offload the origin servers when they are unable to service the incoming requests due to one of the following reasons:

- Whenever a directory with objects is moved from one location to another. In this case, the origin server usually responds with an HTTP 302 redirect message.
- Whenever certain objects are removed. The origin server usually responds with an HTTP 404 "File not found" message.

In the preceding scenarios, the origin servers cannot deliver the requested content and are forced to send a response such as "service unavailable." You can configure Media Flow Controllers to cache these responses for a specific duration in the RAM cache, so that any subsequent request is served from the Media Flow Controller cache rather than from the origin servers. This configuration ensures that the origin servers are not overloaded during their downtime.

You can configure Media Flow Controller to cache 3xx (except 304 response) and 4xx origin server responses. However, you can configure the duration from 60 seconds to 28,800 seconds (8 hours). Media Flow Controller serves any subsequent request for the object from its cache within the specified duration.

The maximum RAM size that can be used for caching non-200 and non-206 responses is limited by the **ram-cache cache-size-mb unsuccessful-responses *<value>*** configuration. When the configured limit is reached, the older responses are evicted. If the eviction fails, then all subsequent responses are tunneled.

If the origin server responds with a 3xx or 404 response to the 200 OK or 206 responses that were previously cached, then the previously cached responses or objects are deleted. With respect to the revalidation path, Media Flow Controller does not cache the new response code in the first attempt. However, if a new request comes for the same URI, then Media Flow Controller might cache the response on the basis of the criteria specific to the non-200 or non-206 response.

NOTE: **Limitations and Constraints:**

- The 5xx responses are not cached.
- If non-2xx responses are chunked, then these responses are not cached.
- Non-2xx responses cannot be compressed.
- Revalidation of cached non-2xx responses is not supported.

A sample access log entry when a non-2xx response is cached is as follows: **10.1.1.101 - - [07/Nov/2012:09:58:16 +0000] GET /file_406_data_5120_120 HTTP/1.1 406 5120 Origin 5120 Origin:10.1.1.101_5120_ unsucces_001 test-vos-qa03**. As shown in the log entry, Media Flow Controller sets the value of the "cache-provider (%c)" field to **Origin** or **Buffer** when

non-2xx responses are cached. If this feature is disabled—that is, if you choose to tunnel the non-2xx responses—then Media Flow Controller sets the value of the cache-provider field to **Tunnel**.

Related Documentation

- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

Using namespace delivery protocol http origin-fetch for Cookie Revalidation

Media Flow Controller performs cookie revalidation to maximize bandwidth savings when caching websites that set cookies for users. Web portals set cookies to differentiate user sessions.

A given Web portal may contain:

- Objects that are "user specific," such as user's profile photo and personal data.
- Objects that are "common to all" users, such as website logo, and videos that get shared between users.

But, Web portals set cookies for the entire user session; so, "common to all" objects as well as "user-specific" objects have cookies set. Cookie revalidation improves bandwidth savings when caching large video objects that are " common to all," but have "user specific" cookies set. Media Flow Controller revalidates such objects in the cache, by sending an **If-Modified-Since** request to the origin, before serving the object to other users.

For Media Flow Controller to do this, you must configure **namespace *name* delivery protocol http origin-fetch header set-cookie any cache permit**; by default, this setting is disabled.

The following is an example of how this setting allows Media Flow Controller to do cookie revalidation:

1. Jack requests **http://www.portal.com/cookievideo.flv** (through Media Flow Controller).
2. Media Flow Controller does not have the object in cache and so sends the request to the configured origin server.
3. The origin server sets a cookie, **Set-Cookie: Jack**, and sends the response to Jack.
4. The origin server also sends the **Cache-Control: must-revalidate** or **Cache-Control: proxy-revalidate** header.
5. Media Flow Controller caches the content with the cookie, **Set-Cookie: Jack**.
6. Jill requests **http://www. portal.com/cookievideo.flv** with **Cookie: Jill**, set in the user browser due to a previous session.
7. Media Flow Controller finds that the object is in the cache, but with **Cache-Control: must-revalidate** header.
8. Media Flow Controller revalidates Jill's request with the origin server:
   a. Media Flow Controller sends the request to the origin server with Jill's cookie, **Cookie: Jill**, and the **If-Modified-Since** header for revalidation
   b. The origin server returns "304 not modified" (if the content can be served to the user as-is).
   c. The proxy serves the same content in the cache to Jill.
9. If the origin server returns **200 OK**, Media Flow Controller serves the content received from the origin server to Jill.

NOTE: **If the response has a "Set-Cookie" header and the required namespace configuration (delivery protocol http origin-fetch header set-cookie any cache permit) is not configured, Media Flow Controller tunnels the response.**

Related Documentation

- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

Using namespace delivery protocol http origin-fetch redirect-302

For pure reverse proxy namespaces only, 302 redirect support is available using the **namespace** HTTP **origin-fetch redirect-302 handle** option. If Media Flow Controller receives an HTTP 302 response, it checks for the Location header in the response message and takes an action:

- If no Location header is present, send the response back to the client.
- If the Location header is present, extract the value (an absolute URI or a relative path).

- If the Location header has an absolute URI value with optional components of port and query parameters, connect to the origin specified by the Location header, and send the request to origin. For example, all of these requests would be redirected:
  - Location: http://www.example.com:80/index.htm
  - Location: http://www.example.com/index.htm
  - Location: http://www.example.com/index.html?count=10
  - Location: https://www.example.com/index.htm
- If the Location header has a relative path, then derive the hostname or domain name and port number (which is optional) from the original request to form an absolute URI and redirect the request to the new origin server specified by the Location header. For example, all of these requests would be redirected:
  - Location: /d1/d2/ (old URI: /a1/b1/f, new URI: /d1/d2/f)
  - Location: d1/d2/ (old URI: /a1/b1/f, new URI: /a1/b1/d1/d2/f)
  - Location: /d1/f1 (old URI: /a1/b1/f, new URI: /d1/f1)
  - Location: d1/f2 (old URI: /a1/b1/f, new URI: /a1/b1/d1/f2)
- If query parameters are set in the original request and the associated **namespace** is configured to allow caching of objects with query parameters, construct the new URI by appending the query parameters to the URI specified in the Location header.
- If the Set-Cookie header is present in the 302 response from the origin, do not redirect; instead, tunnel the response to the client.
- If the Cookie header is present in the client request, do not redirect.
- If the object is not cacheable, tunnel it back to the client. If the object is cacheable, attempt to cache the object (subject to configured caching constraints).

For subsequent requests for the original URI, Media Flow Controller responds with the cached object. Object revalidation is attempted by putting a request to the original server, instead of revalidating from the server that responded with a non-302 response. If the original server continues to respond with a 302, then the Media Flow Controller follows the Location header as outlined above and revalidates the content from the new origin.

Other HTTP response messages such as 300, 301, 307 are not cached and are tunneled to the client.

The namespace HTTP **origin-fetch redirect-302 passthrough** setting causes all 302 responses from the origin to be tunneled back to the client.

Related Documentation

- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

Caching Origin Fetch Requests Using Tunnel Override

By default, the system does not cache content when an origin request is sent with cache control set to no transform or when the origin server sends expired objects. See "Factors Influencing Whether an Object Will Be Cached" on page 13.

This bypass mechanism is called "tunneling." For transparent proxy deployments, the main benefit of cache tunneling is that by overriding certain tunneled transactions, the cache hit ratio is improved for the remaining transactions. All of the client request tunnel-override options are disabled by default.

All of the origin fetch tunnel-override options are disabled by default. To enable the origin fetch tunnel-override options, use the **namespace *name* delivery protocol http origin-fetch tunnel-override cache-control-no-transform** and **object-expired** CLI commands. See the "(namespace) delivery protocol http origin-fetch" section in the *Media Flow Controller CLI Command Reference*.

Caching Origin Fetch Cache Control Requests

The system does not cache origin fetch requests that are set to Cache-Control: no-transform by default.

To enable origin fetch cache control no transform requests:

1. Enter the CLI configuration mode.

   > **enable**

   # **config terminal**

2. Enable the cache-control-no-transform tunnel-overide option.

   (config) # **namespace name delivery protocol http origin-fetch tunnel-override cache-control-no-transform**

   The system caches origin fetch responses with the cache control header set to no-transform Tunnel_77.

3. View the tunnel-override cache-control-no-transform configuration.

   (config) # **show namespace *name***

   ```
   Origin Fetch Configuration:
     Byte Range : preserve (aligned)
     Cache-Age (Default): 0 (seconds)
     Cache-ingest Hotness Threshold: 3
     Cache-ingest Size Threshold: 0
     Cache-Fill : Client-Driven
     Client-Driven Aggressive Threshold: 9
     Cache Age Threshold: 60 (seconds)
     Cache-Directive: follow
     Object Size Threshold: 4096 Bytes
   ```



```
Object Size Minimum Threshold: 0 (KiB)
Object Size Maximum Threshold: 0 (KiB)
Modify Date Header: deny
URI Depth Threshold: 10
Treat End of Transaction on connection close (partial-fetch): No
Custom Cache Control Header: None
Redirect-302 : Pass-through
Cache Set-Cookie Header : Deny
Packing Policy type : lifo
Tunnel-Override Configuration:
Cache-control-no-transform
Object Correlation Etag Ignore : No
Object Correlation Validators-all Ignore : No
```

Caching Origin Fetch Object Expired Requests

By default, the system does not the cache the following origin fetch responses:

- Ones that indicate expired objects with a new expiration date set to current
- Ones with cache_age default configured to Tunnel_87.

To enable caching of origin fetch object expired requests:

1. Enter the CLI configuration mode.

   > **enable**

   # **config terminal**

2. Enable the object-expired tunnel-override option.

   (config) # **namespace name delivery protocol http origin-fetch tunnel-override object-expired**

   The system caches origin fetch responses that indicate expired objects, and sets a new expiry date as the current date plus the configured cache_age default. The tunnel reason code for such responses will be set to Tunnel-87 in access logs.

3. View the tunnel-override object-expired configuration.

   (config) # **show namespace *name***

   ```
   Origin Fetch Configuration:
     Byte Range : preserve (aligned)
     Cache-Age (Default): 0 (seconds)
     Cache-ingest Hotness Threshold: 3
     Cache-ingest Size Threshold: 0
     Cache-Fill : Client-Driven
     Client-Driven Aggressive Threshold: 9
     Cache Age Threshold: 60 (seconds)
     Cache-Directive: follow
     Object Size Threshold: 4096 Bytes
     Object Size Minimum Threshold: 0 (KiB)
     Object Size Maximum Threshold: 0 (KiB)
     Modify Date Header: deny
     URI Depth Threshold: 10
     Treat End of Transaction on connection close (partial-fetch): No
   ```

```
Custom Cache Control Header: None
Redirect-302 : Pass-through
Cache Set-Cookie Header : Deny
Packing Policy type : lifo
Tunnel-Override Configuration:
Object-expired
Object Correlation Etag Ignore : No
Object Correlation Validators-all Ignore : No
```

Related Documentation

- Caching Client Requests Using Tunnel Override on page 83
- Factors Influencing Whether an Object Will Be Cached on page 13

Using namespace domain regex

This section provides some examples of **namespace domain regex** use. Change specifics accordingly.

NOTE: Regex entries do not contain spaces. Also make sure to enclose all regex entries in double quotation marks (as shown in the examples).

See Table 8 on page 95.

Table 8: Example namespace domain regex Entries

Regex	Example Matches
"www.example.com|example.com"	www.example.com example.com
.*example.com	
"^[a-f,0-9]{8}\.(origin\.|cdn\.)?cms\.example\.com:80$"	abcdef02.origin.cms.example.com:80 abcdef23.cdn.cms.example.com:80 abcdef09.cms.example.com:80
"^cms[0-9]{3}\.(dc2|qcg7)\.example\.com:80$"	cms123.dc2.example.com:80 cms079.qcg7.example.com:80
"^orig.(sv1|qcg1|qcg5)\.example\.com:80$"	orig.sv1.example.com:80 orig.qcg1.example.com:80 orig.qcg5.example.com:80
"^(cms[0-9]{3}).*(qcg[0- 9]+|sv1|ch1|dc2|af1)\.example\.com:80$"	cms123.x.y.qcg0.example.com:80 cms257.x.y.qcg01.example.com:80 cms379.x.y.sv1.example.com:80 cms222.x.y.ch1.example.com:80 cms876.x.y.dc2.example.com:80 cms343.x.y.af1.example.com:80

Related Documentation

- Creating a Namespace and Setting Namespace Options (CLI) on page 76

- Media Flow Controller Namespaces on page 16

Using namespace domain <FQDN:port>

Media Flow Controller can listen for requests on multiple TCP ports, up to 64. The default is port 80 for HTTP, and port 554 for RTSP. To map incoming requests to the correct namespace, especially on a non-default port, the **namespace domain** must be set properly with port number included. Media Flow Controller maps an incoming URL to a namespace by extracting the value against the HOST header and matching it to the value configured for the **namespace domain**.

When non-default port numbers are used, you must ensure that the HOST header has the port number coded correctly in the incoming URL and in the **namespace domain**.

For example, if Media Flow Controller listens on ports 80, 8080, and 4040 for incoming HTTP requests, and the requests on port 80 must go to namespace **ns80**, requests coming in on port 4040 must go to namespace **ns4040**, and requests on port 5050 must go to namespace **ns5050**. The configuration would be as follows:

```
namespace ns80
  domain video.example.com
namespace ns8080
  domain video.example.com:4040
namespace ns4040
  domain video.example.com:5050
```

For requests to match **ns80**, **domain**/HOST, the header must be **video.*example*.com**.
For requests to match **ns4040**, **domain**/HOST, the header must be **video.*example*.com:4040**.
For requests to match **ns5050**, **domain**/HOST, the header must be **video.*example*.com:5050**.

Related Documentation

- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

Using namespace precedence

Use **namespace *name* precedence** to set unambiguous mapping of incoming GET requests in the case of **match *criteria*** overlap.**precedence** is set at the namespace level during namespace creation. If no precedence is set, the precedence value is set by default to the highest current precedence plus 10. The lower the number, the higher the preference for that namespace; values **11** (highest precedence) through **65535** (lowest precedence) can be used. For example, consider three URLs and namespaces as follows:

- http://a.com/abc/def/file1.flv
- http://a.com/abc/file2.flv
- http://a.com/pqr/file3.flv

```
namespace ns1
  domain a.com
  match uri /abc/def
  precedence 1
  origin-server http o1.com
  status active
  namespace ns2
  domain a.com
  match uri /abc
  precedence 2
  origin-server http o2.com
  status active
  namespace ns3
  domain a.com
  match uri /
  precedence 3
  origin-server http o3.com
  status active
```

All three URLs match namespace **ns3** with **domain** (**a.com**) and **match uri /** (slash). To ensure that **match uri** #1 (**/abc/def**) maps to **ns1** and not **ns3**, set the **precedence** value. Likewise, for **match uri** #2 (**/abc** to map to **ns2**, a **precedence** value is needed. Only **match uri** 3 (**/**) should be mapped to **ns3** with the **precedence** value configured as shown.

NOTE: Excessive use of **precedence** has performance impact as **precedence** allows for longest prefix matching. If possible, configure namespaces so that they have no overlaps in the **domain** and match *criteria* combination, which are used for mapping the incoming HTTP GET to a namespace.

Related Documentation

- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

Using namespace match uri regex

This section provides some examples of **namespace match uri regex** use.

NOTE: Regex entries do not contain spaces. Also make sure to enclose all **regex** entries in double quotation marks (as shown in the examples). See Table 9 on page 97.

For **namespace match uri** *uri-prefix*, the regex is written against the absolute path portion of the URL. For example, given the URL **http://abc.com:8080/index.html**, **/index.html** would be the absolute path portion of the URI.

Table 9: Example namespace match uri regex Entries

Regex	Example Matches
"\/A\/b\/c\/d\/index\.html"	/A/b/c/d/index.html

Table 9: Example namespace match uri regex Entries *(continued)*

Regex	Example Matches
"\/A\/b\/c\/d\/index\..*"	/A/b/c/d/index.html /A/b/c/d/index.html
"\/A\/b\/.*\/d\/index\..*"	/A/b/x/d/index.html /A/b/xx/d/index.html /A/b/abc/d/index.html

Related Documentation

- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

Using namespace match virtual-host

You can specify a virtual host for the **namespace match** criteria.

To do this, enter the IP address of the virtual host and, optionally, a port number. You can also set a precedence, if needed. In this configuration, the destination IP of the incoming session is used to find the match to the namespace. For the match to be based only on the destination IP, set **domain** to **any**. In that way, the incoming session destination IP and destination port (if given) are used to match to the namespace. For example, from namespace prefix mode:

- Match all sessions with destination IP10.1.1.1 to this namespace:

  match virtual-host 10.1.1.1

- Match all incoming requests with destination IP 10.1.1.1 and port 8080 to this namespace:

  match virtual-host 10.1.1.1:8080

- Match all incoming requests on destination port 8080 to this namespace:

  match virtual-host 0.0.0.0:8080

Related Documentation

- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

Using Namespace Object Compression

- HTTP Object Compression Overview on page 99
- Configuring HTTP Object Compression on page 100
- Verifying HTTP Object File Compression on page 103

HTTP Object Compression Overview

Content providers typically publish uncompressed content to Content Delivery Networks (CDNs). CDNs optimize bandwidth usage at the provider edge and improve user experience by serving compressed content to clients, provided the Web browser is capable of handling compressed content. HTTP object compression is very useful when delivering static web content to mobile users. Compression is essential in Reverse Proxy deployments. Clients fetch compressed content using HTTP and HTTPS protocols.

Supported Web Browsers

Web browsers, such as Internet Explorer, Mozilla Firefox, Google Chrome, and Opera, that are compliant with RFC 2616, are capable of handling compressed content. A browser expresses its ability to handle compressed content by using the standard Accept-Encoding HTTP header.

Supported Object Compression File Types

The decision to compress an object is made based on the namespace configuration. Media Flow Controller compresses static, plain text objects, such as txt, html, htm, js, and css. See "Configuring HTTP Object Compression" on page 100.

Object Compression Modes

There are two modes for compression based on the object size:

- Offline compression, for objects of size greater than 32 Kb.
- Inline compression, for objects of size less than 32 Kb.

Offline Compression—When a client requests an object greater than 32 Kb that is not in cache and the namespace has been configured for compression, the response from the origin server is tunneled to the client. Media Flow Controller initiates an offline fetch of the object, compresses it, and saves it to the cache file system. For offline compression, the system tunnels the object for the first client request and initiates an offline fetch of the content from the origin server. Offline fetch is initiated only when the object becomes ingest worthy. The purpose of offline compression is to compress objects without affecting the user experience for the first user who requested the content.

Inline Compression—When a client requests an object less than 32 Kb that is not in cache and the namespace has been configured for compression, the response from the origin server is cached, compressed inline, and passed on for caching. The compressed object

can stay only in the RAM cache. The object is ingested into the disk cache only when it becomes popular. The object might be evicted from RAM cache if it doesn't become popular.

Supported Compression Methods

Objects are compressed using Gzip and deflate compression methods. The decision to use Gzip versus deflate is based on the browsers or clients to which the content is being served. Gzip is the most popularly used compression method and is supported by many popular web browsers on the market.

Compressed Object Behavior

Compressed objects inherit the headers such as Expiry, Last-Modified-Date, Date, and so on, from the original uncompressed form of the object. When Media Flow Controller revalidates the object with the origin server, the revalidation happens with the uncompressed object URL and headers.

The compressed object is served to the clients when the client sends an Accept-Encoding header, expressing the desire to accept compressed objects.

The byte-range requests for a compressed object in cache are tunneled, like any other request.

Chunked responses are compressed offline and stored as compressed.

Media Flow Controller stores only one version of the object in the cache: either the compressed form or the uncompressed form. Once an object is compressed and stored in the cache, requests from clients for an uncompressed object are tunneled to the origin server.

Purging Compressed Objects

Requests to purge the content occur using the original URL of the uncompressed form of the content. To purge compressed objects, use the **namespace object delete** CLI command.

Monitoring Object Compression

Counters track compressed object performance per namespace, including network bandwidth saved due to compression. The access log indicates when compressed content was served. The **show namespace *name* object list** CLI command displays objects in the cache (including the compressed objects) and indicates whether an object is compressed or not. See "Verifying HTTP Object File Compression" on page 103.

Related Documentation

- Configuring HTTP Object Compression on page 100
- Verifying HTTP Object File Compression on page 103

Configuring HTTP Object Compression

You configure object compression parameters per namespace using the **namespace *name* object compression** CLI command options. You can enable or disable object

compression, specify one or more file types to compress, specify the object minimum and maximum size range for compression in bytes, and specify the compression method.

Enabling and Disabling HTTP Object Compression

By default, file compression is disabled.

To enable or disable object compression:

1. Enter the CLI configuration mode.

   > **enable**

   # **config terminal**

2. Enable or disable HTTP object compression.

   a. To enable object compression:

      # **namespace *name* object compression enable**

   b. To disable object compression:

      # **namespace *name* object compression disable**

Configuring the HTTP Object Compression File Type

You can configure the system to compress one or more file types, such as txt, .js, .html, .htm, and .css.

1. Enter the CLI configuration mode.

   > **enable**

   # **config terminal**

2. Specify the file types to compress.

   You can specify several file types at once by separating them by a comma (,).

   For example, **namespace *name* object compression file-types *.txt, .js, .html, .htm, .css***

Configuring the HTTP Object Compression Size Range

The system compresses objects ranging from a minimum of 128 bytes to a maximum of 10485760 bytes (10 GB). Objects less than 32 KB or 32768 bytes are compressed inline and cached. For objects greater than 32 KB, the system performs an offline fetch of the object, compresses it, and saves it to the cache file system.

To specify the object compression size range:

1. Enter the CLI configuration mode.

   > **enable**

   # **config terminal**

2. Specify the object size threshold for compression.

   # **namespace name object compression size-range** ***min_obj_size_in_bytes max-obj-size-inbytes***

   The range values must be between 128 (or above) and 10485760 (up to 10 MB).

   For example, **namespace name object compression size-range** ***1024 32000*****.**

Configuring the HTTP Object Compression Method

The system compresses objects using either the Gzip or deflate object compression method. The decision to use Gzip versus deflate is based on the browsers or clients to which the content is being served. Gzip is the most popularly used compression method and is supported by many popular web browsers on the market.

To specify the object compression method:

1. Enter the CLI configuration mode.

   > **enable**

   # **config terminal**

2. Specify the type of compression:

   # **namespace** ***name*** **object compression algorithm (gzip | deflate)**

3. Exit configure CLI mode.

   # **exit**

Purging Compressed Objects

Requests to purge the content are supported using the original URL of the uncompressed form of the content.

To purge compressed objects:

1. Enter the CLI configuration mode.

   > **enable**

```
# config terminal
```

2. Purge the compressed objects that you want.

   Use the **namespace *name* object delete URI|pattern|all** CLI command. A URI or pattern removes objects matching the uri or given pattern. All removes objects in the cache associated to the namespace.

Related Documentation

- HTTP Object Compression Overview on page 99
- Verifying HTTP Object File Compression on page 103

Verifying HTTP Object File Compression

You can view the object compression configuration settings for a namespace. You can verify that a compressed object is in the cache. Additionally, you can view the access log to ensure that a compressed object has been served. The system provides counters for the namespace to monitor the number of objects compressed, bandwidth savings, and the number of compressed objects served.

- Viewing the Namespace Object Compression Configuration on page 103
- Monitoring Object Compression Counters on page 104
- Verifying that Compressed Content Is Served on page 104
- Verifying Compressed Objects in Cache on page 105

Viewing the Namespace Object Compression Configuration

Purpose Monitor the object compression configuration for a specific namespace to ensure appropriate for object compression operation and performance.

Action To view the namespace object compression configuration:

1. Enter the CLI configuration mode.

   ```
   > enable
   # config terminal
   ```

2. View the namespace object compression configuration.

   ```
   # show namespace name
   ```

   ```
   Compression Settings:
     Compression Enable: yes
     Max-Object-Size: 1048 (B)
     Min-Object-Size: 128 (B)
     Compression Algorithm: gzip)
     File-Types:
      .css,    .htm,    .html,
      .txt,
   ```

3. Exit the CLI configuration mode.

   ```
   # exit
   ```

Monitoring Object Compression Counters

Purpose Monitor counters for object compression operation and performance for a namespace, including:

- The number of compressed objects served for from a namespace.
- The amount of network bandwidth saved due to compression.
- Total number of compressed objects per namespace.

Action To view the object compression counters for a namespace:

1. Enter the CLI configuration mode.

   > **enable**

   # **config terminal**

2. View the compression counters.

   # **show counters internal**

Table 10: Namespace Object Compression Counters

Field	Description
ns.*namespace_name*.http.client.compress_cnt	The number of compressed objects served from a namespace.
ns.*namespace_name*.http.client.compress_bytes_save ns.*namespace_name*.http.client.compress_bytes_overrun	These two counters are used to determine the number of bytes saved due to compression (such as, bytes saved equals (=) ns.*namespace_name*.http.client.compress_bytes_save — (minus) ns.*namespace_name*.http.client.compress_bytes_overrun).
ns.*namespace_name*.http.server.compress_cnt	Total number of compressed objects per namespace.

3. Exit the CLI configuration mode.

   # **exit**

Verifying that Compressed Content Is Served

Purpose Monitor the access log to ensure that compressed content is served to the client.

Action

1. Enter the CLI configuration mode.

   > **enable**

   # **config terminal**

2. View the access log contents.

   For example:

   ```
   Buffer 10.157.42.57 10.157.42.213 -
   [11/May/2012:06:08:02 +0000] GET /5MB HTTP/1.1 206 34
   [G/D] fproxy - - - - - 0 kd-mfc
   ```

   Where **G** indicates gzip compression, **D** indicates Deflate compression, and a "-" indicates that the object is not compressed.

3. Exit the CLI configuration mode.

   # **exit**

Verifying Compressed Objects in Cache

Purpose Display objects in the cache (including the compressed objects) and shall indicate whether an object is compressed or not.

Action To view cached compressed objects:

1. Enter the CLI configuration mode.

   > **enable**

   # **config terminal**

2. Enter # **show namespace** *name* **object list**

   ```
   ---------------------------------------------------------------------------------
   * Loc  Pinned  Compressed Size (KB)       Expiry                    URL
   *---------------------------------------------------------------------------------
     RAM   N     G 2016  Fri May 11 07:23:04 2012
   10.157.42.213:80/669db18ba8f93e8cf574e5b2aedaeeba
   ```

   Where **G** indicates gzip compression, **D** indicates Deflate compression, and a hyphen (-) indicates no compression.

3. Exit the CLI configuration mode.

   # **exit**

Related Documentation

- HTTP Object Compression Overview on page 99
- Configuring HTTP Object Compression on page 100

Using namespace object delete | list | revalidate

You can list, delete, or revalidate contents in a namespace. The command uses the name of a **namespace** and applies a **list**, **delete**, or **revalidate** operation to the objects matching the given pattern. Use **revalidate all** to validate the contents of the cache present under the namespace. From the time when issued, Media Flow Controller serves the content to the client only after revalidating the content with the origin.

NOTE: Currently, the MFC supports only the "*" and "?" shell meta characters. It does not support range or group meta characters.

namespace *name* object (list |delete | revalidate) (all | *URI* | pattern) [*domain*] [*port*]

NOTE: The domain and port options only apply to namespaces with proxy-mode mid-tier or proxy-mode virtual. This way, objects cached for a given domain or domain and port are listed instead of all objects irrespective of domain.

For example, with this namespace and a URL of **http://example.com/abc/def/file.flv**:

```
namespace ns1
  domain example.com
  match uri /abc
```

- To list an object and get its characteristics:

  ```
  namespace ns1 object list /abc/def/file.flv
  ```

- To delete an object with the same URL:

  ```
  namespace ns1 object delete /abc/def/file.flv
  ```

- To delete all objects in that namespace's disk cache with the same URL:

  ```
  namespace ns1 object delete all
  ```

- To list the first 50 objects in that disk cache and create a file named with the UUID of the namespace listing all cached objects for that namespace:

  ```
  namespace ns1 object list all
  upload object list namespace SCP
  ```

  In the example, if the namespace had a UUID of **80213A2C**, the file containing the list is **80213A2C.lst**. Only the first 50 cached objects are listed; if there are more than 50, use the **upload** command.

- To list and delete objects based on patterns (for example, you can specify ***.flv** as a pattern)

  Media Flow Controller does not support a full regular expression for deleting or listing. An object list command with a wildcard must be prefaced with at minimum the URI specified in the namespace match uri configuration. For example, if the namespace was constructed using this command:

namespace *foo* match uri */abc/def*
and then you issue the following list command:
namespace*foo* object list /*
the result would be zero objects, because the namespace starts with ***/abc/def***. You would have to use either of the following two CLI commands to see all objects:
namespace *foo* object list /abc/def/*
or
namespace *foo* object list all

- URIs with space in the URI name cannot be deleted or listed by providing the exact URI as the argument for the namespace list or delete command. Always use a wildcard to cover up spaces embedded in the URI name while listing and deleting the URI.

Related Documentation

- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

Using namespace object validity-begin-header

Use the namespace *name* object validity-begin-header *header* option to indicate the earliest time a particular object can be requested by a user. Any user request for the object before the validity start time will be responded back with "404 File not found" message. Currently, the “VAL-BEGIN” header is the only acceptable value for validity-begin-header. With this option configured, the expected response header from the origin server would look like this:

```
VAL-BEGIN: Wed, 18 Dec 2010 04:58:08 GMT
```

Which would imply that this object will be served only after Wed,18 December 2010.

Related Documentation

- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

Using namespace origin-server http idle-timeout

You can configure the origin idle-timeout value at an individual namespace level. This feature allows the administrator to assign additional timeout values for those websites that are expected to take longer to respond to requests, such as websites that must process lengthy database queries prior to responding.

To configure this feature, enter the following command:

namespace *namespace* origin-server http idle-timeout *seconds*.

Using Namespace for Pre-Staging Content with FTP

When creating a namespace for pre-staging the content using FTP, be sure to configure the **namespace <*name*> delivery protocol http origin-fetch cache-age-default *seconds*** to a large, non-zero value; for example, **28800**.

NOTE: The FTP ingest interface has been qualified for use only in conjunction with the Juniper Mobile Video Optimization solution. Other use cases are not currently supported.

Related Documentation

- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

Using Namespace for Live Streaming Delivery Without Caching

The following is an example **namespace** configuration to deliver live streaming objects without caching. Use the **delivery protocol** or **live-pub-point** command to enter prefix mode.

```
namespace name
  match uri uri-prefix
  origin-server rtsp IP_address | hostname [port]
  status active
  delivery protocol rtsp
  exit
  live-pub-point pp_ name
  receive-mode on-demand
  status active
  exit
  exit
```

Related Documentation

- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

Using Namespace for Live Streaming Delivery with Caching

The following is an example **namespace** configuration to deliver live streaming objects with caching. Use the **delivery protocol** or **live-pub-point** command to enter prefix mode.

```
namespace name
  match uri uri-prefix
  origin-server rtsp IP_address | hostname [port]
  status active
  delivery protocol rtsp
  exit
  live-pub-point pp_ name
  receive-mode on-demand
  status active
  caching enable
  exit
  exit
```

Related Documentation

- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

Using Namespace for Proxy Configurations

You can use **namespace** settings to configure Media Flow Controller to operate as a proxy in various ways:

- **Reverse proxy**—Caches and delivers content for a set of domains; client requests are routed to a configured IP address. Setting **namespace origin-server** to ***FQDN*** or **server-map** implies a reverse proxy configuration. Media Flow Controller as an edge cache is effectively a **reverse** proxy that reduces network and CPU load on an origin server by serving previously retrieved content, and enhances user experience by decreasing latency. See also "Reverse Proxy Deployment Process" on page 450.
- **Mid-tier proxy**—Caches and delivers content for a set of domains; client requests are routed to a configured IP address. Setting **namespace origin-server** to **absolute-url** implies a mid-tier proxy configuration. As a **mid-tier** proxy, Media Flow Controller must be explicitly configured in the browser to intercept all requests. After Media Flow Controller receives traffic from the client, it separates the traffic. Cacheable requests are sent using Media Flow Controller for performance-enhanced delivery. Non-cacheable requests are tunneled. See also "Mid-Tier Proxy Deployments" on page 477.
- **Transparent proxy**—Caches popular content and optimizes the backhaul network utilization. The cache is made to look transparent by spoofing the origin server IP address in the response to the client and spoofing the client IP address in the request to the origin. A transparent proxy does not require browser configuration and is not readily visible to end users. As a **transparent proxy** where origin server access is derived from the HOST header, the X-NKN header, or the destination IP address given in the incoming request, explicit **origin-server** configuration is not possible. Use this as an alternative to providing a single origin server address. Be sure that **delivery protocol http allow-req** is set to **all** (the default).

Table 48 on page 465 gives details about the configurations and defaults per proxy deployment.

Table 11: Namespace origin-server and origin-request Dependencies per Proxy Mode

Proxy Mode	origin-server Setting Default for origin-request host header inherit incoming-req	For Origin Use the...	Source IP for Cache Miss	Destination IP for Cache Miss
Reverse	**http** ***FQDN***	Specified FQDN	Media Flow Controller IP address	DNS resolved IP address of FQDN
	deny			
Reverse	**http server-map**	Server-map HTTP mount point	Media Flow Controller IP address	DNS resolved IP address of server-map origin-server
	deny			

Table 11: Namespace origin-server and origin-request Dependencies per Proxy Mode *(continued)*

Proxy Mode	origin-server Setting Default for origin-request host header inherit incoming-req	For Origin Use the...	Source IP for Cache Miss	Destination IP for Cache Miss
Reverse	**nfs** ***FQDN:path***	NFS mount point	None (NFS mounted)	None (NFS mounted)
	permit			
Reverse	**nfs server-map**	Server-map NFS mount point	None (NFS mounted)	None (NFS mounted)
	permit			
Mid-Tier	**http absolute-url**	Client request absolute URL	Media Flow Controller IP address	DNS resolved IP address of origin server chosen from the client request
	permit			
http follow header ***name*****Transparent** (origin based on HOST header)	**http follow header HOST**	HOST header value	Media Flow Controller IP address	DNS resolved IP address of the origin server from the HOST header
	permit			
Transparent (origin based on X-NKN or custom header)	**http follow header** ***name*** **use-client-ip**	Specified header	Client IP address	DNS resolved IP address of the origin-server from the given header
	permit			
Transparent (origin based on client destination IP)	**http follow dest-ip**	Client request destination IP	Media Flow Controller IP address	No DNS resolution. Origin IP is client destination IP
	permit			
Transparent	**http follow dest-ip use-client-ip**	Client request destination IP	Client IP address	No DNS resolution. Origin IP is client destination IP
	permit			

CAUTION: If a non-default **origin-request host header inherit incoming-req** value is configured against an **origin-server** setting, the behavior is undefined and no error or warning is issued. For example, if **origin-server http absolute-url** is set (Mid-Tier proxy), and you set **origin-request host-header inherit incoming-req deny**, the behavior is undefined.

Related Documentation

- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

Example: Configuring Media Flow Controller Namespaces (CLI)

Configure namespaces to set fine-grained delivery policies. Every Media Flow Controller deployment must have at least one namespace, and usually has several.

To configure a namespace:

1. Configure a namespace with a **name** (puts you in namespace configuration mode; enter **exit** when finished, and optionally inherit another namespace's cache or UUID. Use **show namespace list** to find namespace UUIDs.

   namespace *name* [cache-inherit *namespace:UUID*]

2. Configure **domain** settings (the default is **any**). The domain you enter should match what you configured as the HOST header, unless you are using **regex**. You can also append a port number, if needed (and used in HOST header).

   domain (any | regex *regex* | *FQDN*)

3. Configure **origin-server** settings (the example uses **http**). You can configure multiple origin servers with the **server-map** option; **port** specification is optional. See **(namespace) origin-server** in the *Media Flow Controller CLI Command Reference* for CLI details.

   origin-server
   http (absolute-url | follow (header *header* [use-client-ip] | dest-ip [use-client-ip]) |
   server-map *map_name*| *FQDN/path* [*port*])
   nfs (*FQDN:export_path* [*port*] | server-map *name*)
   rtsp (*FQDN* [*port#*]) [alternate *string* [*port#*]]

   TIP: If unsure what port your **origin-server** is using, use standard Linux shell commands (for example, **netstat –nl**) to figure out the port, and then configure it along with the **origin-server**, if not the default. If you need to change the **origin-server**, or any **namespace** setting, simply enter the new setting.

4. Configure **match** criteria options (determines the URI to cache). All **match** options can utilize the **precedence** argument to break ties when namespaces are defined with the same **match** criteria.

   match
   header (*header* | regex *regex*) [precedence *number*]
   query-string (*name* | regex *regex*) [precedence *number*]
   uri *uri-prefix* | regex *regex*) [precedence *number*]
   virtual-host *IP_address* [*port*] [precedence *number*]

5. Configure **delivery protocol** options. The default **delivery protocol** is **http**. To enable **delivery protocol rtsp**, press Enter after **rtsp**; then set RTSP options.

   a. Configure **delivery protocol http client-request** options and manipulate incoming client requests. See **(namespace) delivery protocol http client-request** in the *Media Flow Controller CLI Command Reference* for CLI details.

      client-request

```
cache-control max-age seconds action serve-from-origin
cache-hit action revalidate-always
cookie action (cache | no-cache)
query-string action (cache [exclude-query-string] | no-cache)
```

b. Configure **delivery protocol http client-response** options. Delete headers in, or add headers to, outgoing responses to client requests. Up to 16 **headers**, including the custom header X-Accel-Cache-Control, can be configured with an **action** value (either **add** or **delete**). If you enter only a header ***name***, the only action allowed is delete; if you enter a header ***name*** and ***name***, the only action allowed is **add**.

```
client-response header name [value] action (add | delete)
```

c. Configure **delivery protocol (http | rtsp) connection concurrent session** limit.

```
connection concurrent session integer retry-after seconds
```

d. Configure **delivery protocol origin-fetch** options; most are for **http**. Only **cache-age-default** is available for **rtsp origin-fetch**. See **(namespace) delivery protocol http origin-fetch** in the *Media Flow Controller CLI Command Reference* for CLI details.

```
origin-fetch
  cache-age (content-type string secs | content-type-any secs)
  cache-age-default seconds
  cache-control header custom_header
  cache-directive no-cache (follow | override)
  cache-fill (aggressive | client-driven)
  content-store media [cache-age-threshold secs ] [object-size bytes]
```

e. Configure **delivery protocol http origin-request** parameters for data requested from the origin. See **(namespace) delivery protocol http origin-request** in the *Media Flow Controller CLI Command Reference* for CLI details. Set **origin-request host-header inherit incoming-req** in accordance with the **origin-server** setting; see Table 11 on page 109. Use **x-forwarded-for** to allow (with **enable**) or disallow (with **disable**) setting the X-Forwarded-For header to the client IP address; the default is **enable**.

```
origin-request
  cache-revalidation
  connect
  header
  host header
  read
  x-forwarded-for
```

Example:

```
MFC (config) # namespace test
MFC (config namespace test) # domain any
MFC (config namespace test) # origin-server http example.com/video
MFC (config namespace test) # match uri / precedence 3
MFC (config namespace test) # delivery protocol http
MFC (config namespace test delivery protocol http) # client-request cookie
action no-cache
MFC (config namespace test delivery protocol http) # client-response header
Location action delete
MFC (config namespace test delivery protocol http) # connection concurrent
session 5000 retry-after 5
MFC (config namespace test delivery protocol http) # origin-fetch
cache-age-default 0
```

```
MFC (config namespace test delivery protocol http) # origin-request
x-forwarded-for enable
MFC (config namespace test delivery protocol http) # exit
MFC (config namespace test) # exit
```

6. Optionally, make **live-pub-point** settings if needed for live streaming:
   - **caching**—Enable caching for this service (default is disabled).
   - **receive-mode**—Set a method for receiving live streaming:
     - **on-demand**—When a request is received.
     - **sdp-name** ***URL***—Use a service delivery protocol (SDP) file to set the live publishing point. The URL can be **scp://...** or **http://...** only. After Media Flow Controller encounters this, it pulls the file from the specified location, and saves it in the file system (not disk cache) so it is available for RTP/RTSP. Optionally, select **immediate** to start as soon as the file is retrieved or enter a **start-time** and, optionally, an **end-time**.
   - **status**—Make the live-pub-point active or inactive.
7. Optionally, add an existing **virtual-player** to the new namespace.

   virtual-player *name*
8. Activate the namespace. Verify configurations with **show namespace** ***name***.

   status active
9. Enter **exit** to leave namespace configuration mode.

```
MFC (config namespace test) # virtual-player test
MFC (config namespace test) # status active
MFC (config namespace test) # exit
```

NOTE: Configuration changes, including a namespace deletion, might not be updated for up to 30 seconds. This is due to a deferred update scheme that requires an HTTP request. An internal probe ensures that such a request occurs at least every 30 seconds.

Related Documentation

- Using namespace domain regex on page 95
- Using namespace domain <FQDN:port> on page 96
- Using namespace precedence on page 96
- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Configuring Server Maps (CLI) on page 232

Using Namespace for Dynamic URI Remapping

The commands for dynamic URI mapping specify that an incoming client request with a URI that matches the configured regular expression (regex) value (**client-request**

cache-index url-match *regex*) is matched to a cache index string (**map-to** *map_string*). The optional **no-match-tunnel** option specifies that if the match fails, the request is tunneled; use this option to have HTTP send the request on the normal path with no **cache-index** mapping if the match fails. With **revalidate-always** configured, if the origin server returns "Object Is Modified," the transaction is tunneled, the object is deleted from the cache, if its time-to-live (TTL) has expired, and the new object is fetched into the cache.

Media Flow Controller supports caching of videos from portals that generate dynamic URLs and for trick-play features such as SEEK. The device supports dynamic URLs and trick-play functionality simultaneously.

See the (namespace) delivery protocol http client-request command in the *Media Flow Controller CLI Command Reference* for CLI details.

NOTE: **You must configure revalidate-always in order for dynamic URI mapping to work.**

Some **url-regex** ***regex*** examples:

1. Regex example with no substring address; only **$0** returned:

   If the configured regular expression is
   '/videoplayback\?.*\&id=[^\&]+.*'
   and if the incoming request URL is
   'http://www.example.com/videoplayback?xxx&id=lxxuuu',
   the matching segment in the URL would be **/videoplayback?xxx&id=lxxuuu**.

2. Regex example with one substring denoted (note the parentheses); **$0** and **$1** returned:

   If the configured regular expression is
   '(/videoplayback\?.*)\&id=[^\&]+.*' ? d '/videoplayback?xxx&id=lxxuuu'
   and if the incoming request URL is
   'http://www.example.com/videoplayback?xxx&id=lxxuuu',
   the matching segments in the URL would be
   /videoplayback?xxx&id=lxxuuu and **/videoplayback?xxx**.

3. Regex example with two substrings denoted; **$0**, **$1**, and **$2** returned:

   If the configured regular expression is
   '(/videoplayback\?.*)\&id=[^\&]+.*' ? d '/videoplayback?xxx&id=lxxuuu'
   and if the incoming request URL is
   'http://www.example.com/videoplayback?xxx&id=lxxuuu',
   the matching segments in the URL would be
   /videoplayback?xxx&id=lxxuuu, **/videoplayback?xxx** and **lxxuuu**.

Some **map-to** ***map_string*** examples using the two substrings regex example:

- `/abc/$1/$2 => /abc//videoplayback?xxx/1xxuuu`
- `/XXX$0/$1/$2 => /XXX/videoplayback?xxx&id=1xxuuu//videoplayback?xxx/1xxuuu`
- `$$$1/$2 => $/videoplayback?xxx/1xxuuu`

- `$$$1$$ => $/videoplayback?xxx$`

Related Documentation

- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

Differentiated Services Code Point Marking and Cache Control Overview

Differentiated Services (DiffServ) provides a mechanism for classifying and managing network traffic and improving quality of service (QoS) for clients. It allows network devices, such as routers and switches, to provide differential services to clients. DiffServ uses the Differentiated Services Code Point (DSCP) field in the IP header for both IPv4 and IPv6 packet classification.

Media Flow Controller sets DSCP bits in the IP packets carrying the HTTP response to the client. The DSCP value you configure is between 0 and 63. You configure DSCP per namespace. The response packets matching the namespace are marked with DSCP values as configured from Media Flow Controller to the client. This feature is disabled by default. When Media Flow Controller receives a request from a user, the TCP connection is associated with a namespace. DSCP values are set based on the configuration in the namespace. After the request is handled, the TCP connection is returned to the global pool and is not be associated to a namespace. Hence, the DSCP value is disabled after handling a given request. For instance, the DSCP value is not set in the IP packets.

Use DSCP marking and cache control to differentiate between namespaces serving videos versus those serving large files or other objects for download. Each can have a separate namespace with a distinct DSCP configuration for setting in client response packets.

To configure DSCP, use the **namespace *name* delivery protocol http client-response dscp *number*** configuration mode CLI command. Use the **show namespace name** or the **namespace details** CLI commands to view the DSCP configuration.

If necessary, use the **no client-response dscp *number*** CLI command to disable the DSCP configuration. To set DSCP values only for a subset of requests handled by a namespace, please refer to Policy Engine chapter.

To set DSCP values only for a subset of requests handled by a namespace, see the Using Media Flow Controller Policy Engine chapter .

Related Documentation

- Configuring Differentiated Services Code Point Marking and Cache Control on page 116
- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

Configuring Differentiated Services Code Point Marking and Cache Control

Differentiated Services (DiffServ) uses the Differentiated Services Code Point (DSCP) field in the IP header for packet classification. You can configure the DSCP field for both IPv4 and IPv6 packets in the HTTP response sent by Media Flow Controller to the user. You configure the DSCP field setting per namespace.

To configure the DSCP:

1. Enter the CLI configure mode.

   # **enable**

   configure terminal

2. Create a namespace or configure an existing one.

   (config) #**namespace** ***name***

3. Configure the namespace delivery protocol.

   (config) #**delivery protocol http**

4. Configure the DSCP field value.

   (config) #**client-response dscp** ***number***

   The DSCP value at the namespace is a decimal value. The DSCP number is a value between 0 and 63. DSCP is disabled by default. For more information about the **namespace delivery protocol http client-response dscp** ***number*** command, see the *Media Flow Controller CLI Command Reference*.

   Use **no client-response dscp** ***number*** to disable the DSCP configuration.

5. Display the DSCP configuration for the namespace.

   (config) #**show namespace** ***name***

   ```
   ...
   Client-Response Configuration :
     DSCP setting: 55
   ...
   ```

Related Documentation

- Differentiated Services Code Point Marking and Cache Control Overview on page 115
- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Media Flow Controller Namespaces on page 16

CHAPTER 4

Configuring Virtual Players

Virtual Players Overview

Virtual players allow Media Flow Controller to overcome the limitations of traditional file based delivery systems by being media aware and allow enforcement of media specific functionality. Virtual players support several highly specialized functions authentication, delivery and storage of media content providing a very high Quality of Experience (QoE) to an end user.

Using virtual players, you can configure highly specialized delivery policies to enable the delivery of different types of media and to control the user experience. These players are capable of interpreting the semantics of incoming requests (for example, query strings), understand the format of the media being requested and apply media specific policies.

For example, a virtual player is can authenticate an incoming request using a signed hash key embedded as a query string in the URL to verify that it is coming from a trusted source. Virtual players can also detect various media file formats and understand their encoding bit rates. The player can subsequently use the information to enforce rate-control to guarantee an optimum transmission rate to ensure a smooth viewing experience to the end user without any buffering. The players also support time and byte based seek that allows a client player to send seek requests to jump to specific portions in the media.

In Media Flow Controller, you can create and assign virtual players to a namespace . You can associate multiple namespaces to a single virtual player if the functionality required by them (for example, namespaces catering to specific web sites or domains) is common. Virtual players have to be configured for web sites that deliver video content. For example,

if you would like to cache and deliver YouTube® video content, create a type YouTube virtual player and associate it to the corresponding namespace setup to serve YouTube requests.

Related Documentation

- Virtual Player Types on page 118
- Virtual Player Features on page 120
- Configuring Rate-Control to Support Video Pacing for Client Players on page 124
- Virtual Player Features on page 120
- Media Flow Controller Virtual Players on page 16

Virtual Player Types

Media Flow Controller supports the following types of virtual players.

- Generic Virtual Player on page 118
- YouTube Virtual Player on page 118
- Yahoo Virtual Player on page 118
- Smoothstream-Pub Virtual Player on page 119
- Flashstream-Pub Virtual Player on page 119
- Virtual Players Feature Support Table on page 119

Generic Virtual Player

The generic virtual player supports most of the options that are commonly required to support video delivery like rate-control, fast-start, seek, authentication, rate-limiting, etc.

YouTube Virtual Player

The YouTube virtual player provides options to:

- Control the rate of media delivery to the player.
- Allow the cache to burst an initial portion of data to quickly fill the client buffer using **fast-start**.
- Support seek via time offsets.
- Interprets YouTube's asset identification mechanism using query parameters.

Yahoo Virtual Player

The Yahoo virtual player supports standard video delivery features along with special health probe and hash verification options.

NOTE: **You must obtain a license to configure Yahoo virtual player features. Juniper Networks will provide you a license key. Use the license install license_key CLI command to apply the license.**

Smoothstream-Pub Virtual Player

The smoothstream-pub virtual player supports publishing Microsoft Smooth Streaming format ISMV files to fragmented format in real time for delivery to Silverlight players without using an IIS Windows Server.

Flashstream-Pub Virtual Player

The flashstream-pub virtual player supports publishing Adobe Zeri or HDS assets to fragmented format for media delivery to Flash player and is similar to the origin module functionality provided by Adobe Flash Media Server (FMS).

Virtual Players Feature Support Table

The features supported by these players are shown in Table 12 on page 119.

Table 12: Virtual Players and Supported Features

Virtual Player Types					
Features Supported	Generic	YouTube	Yahoo	Smoothstream-pub	Flashstream-pub
rate-control	✓	✓	✓		
fast-start	✓	✓			
seek-config	✓	✓	✓		
connection max-bandwidth	✓	✓	✓		
hash-verify	✓				
cache-name		✓			
bandwidth-opt	✓	✓			
full-download	✓				
health-probe			✓		
req-auth			✓		
quality-tag				✓	
fragment-tag				✓	✓
segment-tag					✓
seg-frag-delimiter					✓

Related Documentation

- Virtual Players Overview on page 117
- Virtual Player Features on page 120
- Configuring Rate-Control to Support Video Pacing for Client Players on page 124
- Virtual Player Features on page 120
- Media Flow Controller Virtual Players on page 16

Virtual Player Features

Using Query String Parameters

Query strings are a part of a URL that permits data to be passed from a HTTP client (often a web browser or video player) to a web server or a caching appliance. A typical URL containing a query string is as follows:

```
http://server/path/program?query_string
```

The query part of the URL, is designated with a question mark (?) followed by defined query strings. Query strings are composed of a series of name-value pas where the name is a string with an associated value, of the form below. Additional queries in the URL are separated by ampersands (&).

```
http://server/path/program?param1=value1&param=value2
```

The query-string-parm arguments are, used extensively in defining virtual player configurations in Media Flow Controller. To support features like seek, rate-control, hash-authentication, etc virtual players need to be configured to recognize and interpret the query strings present in an incoming request.

For example, if a client player wants to request a video asset at a particular time-offset (seek) and a specific delivery rate (rate-control), it can do this as follows:

```
http://server/path/video.flv?seek-to=100 &speed=1000
```

Using virtual players, we can pre-configure Media Flow Controller to associate seek requests to the query parameter *seek-to* and specify the units for the value in seconds or milliseconds and delivery-rate to the query parameter speed and specify the units as in Kbit/sec or KBytes/sec. On receiving a request of the above form, Media Flow Controller can interpret it such that it seeks to the 100th second of this asset and then delivers the data at a speed of 1000 Kbps depending on the configured units for each query parameter.

NOTE: In a tiered deployment, the virtual player query-string-parm values you configure in your Media Flow Controller origin server must match the corresponding query-string-parm values configured in your Media Flow Controller edge.

Using Seek Configuration

This feature allows a client player to seek or scrub to a specific part of a video. For example, you can either jump ahead a few minutes or go back a few minutes along the video timeline. The exact position where to seek is provided in the incoming URL using the configured seek-config query parameter. You can provide the seek position using either time or byte offsets. See Table 13 on page 121 for the supported options.

Table 13: Seek Query Parameter Support

Seek/Scrub Support in MFC using Query Parameter Request	Byte Offset	Time Offset (secs/msecs)
FLV file format	Yes NOTE: The seek length option, if configured, only applies to this case.	Yes
MP4 file format	No	Yes

- **Seek Config Query String Parameters**—This option, when configured with the client player seek query string, tells Media Flow Controller which query parameter to look for in an incoming request to support seek operations. Seek is supported for FLV and MP4 file formats via query string parameters.
- **Seek FLV Type**—Select one:
  - **byte-offset**— The value of the seek query-string-parm sent by the client is in bytes. If the seek-length option is desired for FLV content, then set the seek-flv-type to byte-offset.
  - **time-secs**—The value of the seek query-string-parm sent by the client in seconds.
  - **time-msec**—(default) The value of the seek query-string-parm sent by the client in milliseconds.
- **Seek MP4 Type**—Select one:
  - **time-secs**—The value of the seek query-string-parm sent by the client in seconds.
  - **time-msec**—(default) The value of the seek query-string-parm sent by the client in milliseconds.
- **Seek-length query string param**—The value referenced by this query string parameter in a request indicates the number of bytes of data to send starting from the requested seek start position offset. This option, if configured, applies only to FLV files and specifically for those with seek-flv-type configuration in Media Flow Controller set to

byte-offset. For all other file types and seek-types (VIZ, FLV or MP4 with time-offsets), this option has no impact if configured.

- **Tunnel-mode**—Tells Media Flow Controller whether or not to tunnel seek requests to origin server. The default behavior is for Media Flow Controller to serve seek requests from the cache.

NOTE: Enable seek support in Media Flow Controller only when the origin server from which Media Flow Controller fetches content is known to support standard HTTP byte range requests. If the origin server does not support HTTP byte range requests, set the seek configuration to tunnel mode enabled.

NOTE: If you enable seek support in Media Flow Controller, configure the relevant namespace serving these requests Media Flow to handle 302 redirect requests natively.

Seek and scrub requests are usually initiated as a new HTTP request to Media Flow Controller with the desired seek offset (time units) provided as a query parameter value. The query parameter string that carries this value is configured as part of the virtual player configuration to allow Media Flow Controller to detect the seek request. For example, if you use **begin** as a seek query string and a seek request for time offset of 100 seconds is sent to Media Flow Controller, it looks like this:
http://www.domain.com/video/foo.mp4?signature=XYZ&begin=100.

When Media Flow Controller receives this request, it looks in cache to check if the file is present and if the desired offset can be satisfied internally. In case the file is not present or if partially cached, Media Flow Controller has to go to the origin to fetch the delta. When fetching the missing portion from the origin, Media Flow Controller cannot use the begin query string as part of the request to the origin. This causes the origin to generate a new media file with tailored headers for this seek request, which is not cacheable by Media Flow Controller.

To prevent this behavior, Media Flow Controller removes the seek query and instead injects regular byte range headers so that the delta that is fetched from the origin is used to fill the missing data in its cache. Media Flow Controller understands how to translate the seek offset in time to the desired byte offset.

Occasionally, some origin servers may redirect requests to alternate servers using 302 redirect depending on the load. When redirection occurs and Media Flow Controller receives a 302 redirect, its standard mode of operation is to let the 302 redirect flow to the client so that the client can reconnect.

But when the 302 redirect flows to the client for seek requests, the new Location header, as part of the 302 redirect from the origin server, does not have the seek query string (since Media Flow Controller stripped that query when sending the request to the origin), so the client now reconnects to the new location as a regular request. Not having the

seek query string causes a user behavior impact, since all seek requests will just translate to the video to play from the beginning.

To handle this scenario, it is recommended that you enable 302 redirect handling in Media Flow Controller using the following CLI command in configure mode: **namespace** *name* **delivery protocol http origin-fetch redirect-302 handle**)

When you enable 302 redirect handling, Media Flow Controller handles all 302 redirects sent by the origin for the associated namespace, prevents client behavior on seeks, and works as expected when origin servers send 302 redirects.

Using Hash Verification Configuration

Media Flow Controller allows you to validate request URLs before serving content to the user. You can use this feature to allow only authorized users to access cached content and prevent URL replays. Media Flow Controller computes an MD5 hash of a request by combining a part of the URL, specified by the **url-type** and including the **expiry-time-verify** value if used, along with a configured **shared-secret** that is **append**ed or **prefix**ed (as configured) to it. The computed hash digest value is then compared with the hash value provided in the incoming URL using a configured **match query-string-parm**. If a match between the computed and provided hash values is unsuccessful, the request is denied. In addition, a content provider can mark the URL hash as expired using the **expiry-time-verify** function. This prevents users from accessing the same content again without going through the authorization process.

Example URL showing **expiry-time-verify query-string-parm e**, and **match query-string-parm h**:

http://www.example.com/media/foo.flv?e=3312665958&*h=128-bit-md-5-hash*

If Media Flow Controller encounters this URL, and **url-type** is set to **absolute-url**, it takes the entire URL up to the configured **match query-string-parm** (shown as italic font in the example). If **url-type** is set to **relative-uri**, it takes the part of the URL after the access method and domain plus the query string up to the configured **match query-string-parm** (**/media/foo.flv?e=3312665958** in the example). If **url-type** is set to **object-name**, it takes the part of the URL after the last slash plus the query string up to the configured **match query-string-parm** (**foo.flv?e=3312665958** in the example).

The hash value is then computed by either appending or prefixing to the URL (or URL part if **url-type** is set to **relative-uri** or **object-name**) the configured **shared-secret**, and comparing the computed value with the hash value provided with the **match query-string-parm**.

Example if **shared-secret** is appended and **url-type** is set to **absolute-url**:

```
virtual-player name type generic
hash-verify digest md-5 shared-secret string append match query-stringparm h
   expiry-time-verify query-string-parm e url-type absolute-url
```

```
Computed hash value = MD5(http://video.example.com/public/2010/
qwerty.flv?fs=5000&ri=300&rs=1234567 + shared-secret)
```

Example if **shared-secret** is prefixed and **url-type** is set to **absolute-url**:

virtual-player *name* type generic
hash-verify digest md-5 shared-secret *string* prefix match query-stringparm h
expiry-time-verify query-string-parm e url-type absolute-url

```
Computed hash value = MD5(shared-secret + http://video.example.com/public/
2010/qwerty.flv?fs=5000&ri=300&rs=1234567)
```

Example **hash-verify expiry-time-verify** use case if **expiry-time-verify** is set to a non-zero value:

1. The user goes to customer website to play a video.
2. The customer website generates a URL and provides it to the client player. This URL has an expiry time that is secured by hash authentication.
3. The client player uses this URL to fetch content from a video server.
4. The server receives the request, the URL has not expired, server delivers the video.
5. If anyone uses the same URL after some time and sends a request to the server, the server rejects it because the expiry time in the URL will not match the hash expiry time.

The **shared-secret** key is defined on Media Flow Controller and on the client player or browser that generates the video request. The content owner defines a secret key that the client player uses to generate hashed URLs. The content owner configures the same secret key, the **shared-secret**, on the Media Flow Controller so that Media Flow Controller can validate those requests.

NOTE: **Type generic virtual-players have a default match query-string-parm of h; this can be changed in the CLI.**

Related Documentation

- Virtual Players Overview on page 117
- Virtual Player Types on page 118
- Configuring Rate-Control to Support Video Pacing for Client Players on page 124
- Virtual Player Features on page 120
- Media Flow Controller Virtual Players on page 16

Configuring Rate-Control to Support Video Pacing for Client Players

- Rate-Control for Video Pacing on page 124
- Configuring Video Pacing on page 126

Rate-Control for Video Pacing

Media Flow Controller provides efficient video delivery with a video pacing feature that allows you to control the rate of video delivery to a client player with automatic bit-rate detection and bit-rate throttling. Media Flow Controller can detect the encoding bit rate for video file formats such as MP4, FLV, WMV/ASF, WebM and 3GPP (3GP/3G2). If the

rate-control feature is enabled, then Media Flow Controller can enforce the encoding rate during the delivery of media using HTTP Progressive Download (PDL).

Media Flow Controller uses the peak bit-rate value provided in the video file format metadata section as the encoding bit-rate value. For some file formats, this information might not be present in the metadata headers because it maybe an optional parameter. In such scenarios, Media Flow Controller uses the file size and the total play time or duration of the asset to arrive at an average bit-rate value.

During rate enforcement, Media Flow Controller automatically accounts for associated TCP/IP delivery overheads for HTTP progressive downloads. Typically, this delivery overhead amounts to 15 to 20 percent of the video bit rate. The rate-control feature additionally allows administrators to change the rate of delivery by specifying a burst factor value. The default delivery overheadburst-factor value is 1.1, but you can configure the delivery burst-factor value to be from 1.0 to 3 times the detected video bit rate using the CLI.

You can configure a virtual player to enforce video pacing using the rate-control feature with one of these options. The rate-control feature in Media Flow Controller provides a scheme max-bit-rate scheme for the enforcement of pacing.

The **max-bit-rate** option allows a best effort rate control mechanism, whereby if system resources are available, Media Flow Controller delivers at the enforced rate. If the system becomes over provisioned at any point, bandwidth is shared among all active connections. We recommend you use this option for rate control along with proper system provisioning using resource-pools (max bandwidth and number of sessions) to avoid a scenario of oversubscription.

NOTE: **Once the throughput or session capacity of the Media Flow Controller has been met or exceeded, per-session bandwidth served by the Media Flow Controller might be less than the value configured in max-bit-rate.**

MBR Settings and Expected Results

For each of the rate control schemes, options are provided to:

- Auto-detect the bit-rate for known codecs.
- Infer the requested rate control value from an incoming request using a query string parameter.
- Statically enforce a fixed limit for all objects.

An additional option called burst-factor is also provided whereby the speed of video delivery can be increased at a rate greater than the requested or detected rate.

When maximum bit rate (MBR) settings for the network and for the virtual player are configured and auto-rate-detect for the virtual player is turned on, you can expect the resulting download speeds shown Table 14 on page 126 for the different configuration settings.

Table 14: MBR Settings and Expected Download Speed

Case #	Global Network MBR (MBRnw)	Virtual Player MBR (MBRvp)	Virtual Player auto-rate-detect	Expected Download Speed
1	0	0	On	File bit rate
2	0	MBRvp > Bit rate	On	File bit rate
3	0	MBRvp < Bit rate	On	MBRvp
4	MBRnw	MBRvp	On	MBRvp supersedes MBRnw Speed = File bit rate (Case 2) or MBRvp (Case 3)
5	MBRnw > Bit rate	0	On	File bit rate
6	MBRnw < Bit rate	0	On	MBRnw

Configuring Video Pacing

To configure video pacing using a generic virtual player:

1. Create a virtual player of type generic.

   virtual-player <*name*> type generic

2. Configure this virtual player to enable video pacing.

   rate-control max-bit-rate auto-detect burst-factor 1.25

3. Associate this virtual player with a namespace.

   namespace <*name*> virtual-player <*name*>

4. Type **exit** to leave configuration mode.

5. Verify the configuration with **show virtual-player <*name*>**.

For example:

```
# virtual-player vp1 type generic
# #rate-control max-bit-rate auto-detect burst-factor 1.25
# exit
# show virtual-player vp1
  Rate control Configuration
    Enabled: Yes
    Scheme : Max Bit Rate
      Auto-detect:Enabled (ENABLED)
      Static-rate:0 Kbps
      URI-Query:
    Burst-Factor: 1.25
```

Configuring the Generic Virtual Player

Virtual Player Type generic

The **generic** virtual player provides options for nearly all virtual player functions:

- **rate-control**—Provides mechanisms to throttle/pace video delivery to a client player.
- **connection max-bandwidth**—Specifies the maximum allowable bandwidth for a session. The actual session bandwidth used does not exceed this value, even if there is available bandwidth in the link. For a full download, Media Flow Controller tries to allocate this value to the session.
- **fast-start**—Allows the cache to burst an initial portion of media data to quickly fill the client buffer to enable fast-start of video playback. A static size or time period or query string based configuration is provided to enable fast-start using the CLI. The speed of delivery for this portion of data is not limited by any configured **connection max-bandwidth** limits.
- **full-download**—Allows the client to download at the fastest possible speed, up to the configured **connection max-bandwidth**.
- **hash-verify**—Verifies the authorization hash value specified in the URL query string.
- **seek**—Implements seek (for FLV and MP4 media files) based on the value of **query-string-parm**. This function allows the client player to seek a specific part of the media content; for example, to jump ahead a few minutes or go back a few minutes in the video.

NOTE: All features provided by the generic virtual player are optional. If a player is configured without enabling any of its features and associated to a namespace, the player will strip all incoming query requests and cache the object without any query strings.

Configuring Generic Virtual Players (CLI)

The generic virtual player can be used to cache most Web content.

- See the *Media Flow Controller CLI Command Reference* for details on the CLI.
- See for background information.
- Before you configure the Media Flow Controller **virtual-player options**, see

To configure the **type generic** virtual player:

1. Configure a virtual player with a **name** and **type generic** (puts you in virtual-player configuration mode). Use **no virtual-player** ***name*** to delete; use **virtual-player** ***name*** **no**

option to make changes to configurations (either reset to the default or remove the setting).

virtual-player *name* type generic

2. Configure hash verification options. In the current version, only **MDT** digest is supported. Configure a **shared secret** value to be appended or prefixed to the URL as specified, for matching against the hash value provided in the URL and indicated by the **match query-string-parm** you configure. Optionally, configure **expiry-time-verify** and **url-type** to help prevent bandwidth stealing.

   hash-verify digest md-5 shared-secret *string* (append | prefix) match query-string-parm *string* expiry-time-verify query-string-parm *string* url-type *type*

3. Configure download parameters for delivering files at the fastest possible speed. If you set **always**, file downloads are always delivered at the fastest possible speed; otherwise, you must have either a query parameter or a header name that initiate a full download.

   full-download (always | match *string* (query-string-parm *string* | header *header_name*))

4. Optionally, configure **rate-control** delivery optimization. A query parameter can be used or define a static **rate** value or enable using auto-detect. After a value is entered, this parameter is enabled.

   rate-control max—bit-rate { auto-detect | static *value-Kbps* | query-string-parm *string* {kbps|KBps|Mbps|MBps} } [burst-factor *number*]

5. Optionally, configure **connection max-bandwidth** delivery optimization. The default is **0** (unbounded) with the Media Flow Controller license, **200** Kbps without it; you must have the license to change the unlicensed default. Use **no connection** to reset the default.

   connection max-bandwidth (0 | *kbps*)

6. Optionally, configure **fast-start** delivery optimization.

   fast-start (query-string-parm *string* | size *KB* | time (*secs*))

7. Optionally, configure **seek** delivery optimization by specifying a query parameter to find the starting point in the video and playing the video from that point.

   NOTE: By default, seek requests are honored by Media Flow Controller. But an option is provided to tunnel seek requests to a origin server if desired using the **seek-config query-string-parm begin tunnel-mode enable** CLI command.

   seek-config query-string-parm *string* [seek-flv-type (byte-offset | time-secs | time-msec)] [seek-length query-string-parm *string*] [seek-mp4- type (time-secs | time-msec)] [tunnel-mode (disable | enable)]

8. Type **exit** to leave virtual-player configuration mode.

```
MFC (config) # virtual-player test type generic
MFC (config virtual-player test) # hash-verify digest md-5 match query-
  string-parm h shared-secret zpzp prefix
MFC (config virtual-player test) # rate-control max-bit-rate auto-detect
```

```
burst-factor 1.25
MFC (config virtual-player test) # connection max-bandwidth 0
MFC (config virtual-player test) # seek-config query-string-parm sk
seek-flv-type time-sec
MFC (config virtual-player test) # exit
MFC (config) #
```

NOTE: If you enable seek support in Media Flow Controller, configure the relevant namespace serving these requests Media Flow to handle 302 redirect requests natively.

9. Verify configurations with **show virtual-player** ***name***.

Configuring the YouTube Virtual Player

Virtual Player Type youtube

The **youtube** virtual player is designed for YouTube and offers a subset of Media Flow Controller delivery options: **rate-control**, **connection max-bandwidth**, **fast-start**, and **seek** (see"Configuring the Generic Virtual Player" on page 127 for details); and YouTube-specific options for identifying the requested video, **video-id**, and its format, **format-tag**.

Configuring YouTube Video Caching (CLI)

Configure a **youtube** type virtual player if you want to cache and optimize the delivery of YouTube videos.

NOTE: By default, YouTube seek requests are honored by Media Flow Controller. But an option is provided to tunnel seek requests to YouTube origin, if desired. To enable tunneling seek requests to YouTube origin, configure a YouTube virtual player as follows:

```
MFC (config virtual-player ytplayerA)# seek-config query-string-parm begin
  tunnel-mode enable
```

To disable and revert to the original mode of operation, configure a YouTube virtual player as follows:

```
MFC (config virtual-player ytplayerA)# seek-config query-string-parm begin
  tunnel-mode disable
```

Using Virtual Player Type YouTube

YouTube encodes media content using industry standard video and audio compression schemes such as H.264/AVC for video and AAC for audio. It stores the encoded bit streams using FLV, MP4, or 3GP containers, depending on the spatial resolution of the video. Currently, YouTube supports the formats as outlined in Table 15 on page 130.

Associations to one of these formats is signaled through a request originating from the player using a query parameter typically of the type **fmt** or **itag**.

Table 15: YouTube Formats

Media Types	Standard	Medium	High	720p	1080P	Mobile
Format/Tag Values (fmt, itag)	34	18	35	22	37	17
Container Types	FLV	MP4	FLV	MP4	MP4	3GPP
Video Codec	H.264/AVC	H.264/AVC	H.264/AVC	H.264/AVC	H.264/AVC	MPEG-4 Part 2
Audio Codec	AAC	AAC	AAC	AAC	AAC	AAC
Spatial Resolution	320x240 640x480	480x360 480x270	854x480	1280x720	1920x1080	176x144

NOTE: Media Flow Controller can cache HTML5 videos for HTTP progressive downloads. For example, Media Flow Controller supports WebM format for YouTube videos accessed from Chrome or Firefox browsers.

Requests originating from a YouTube player for a video asset typically come in the following two forms:

```
(a) GET "http://www.youtube.com/get_video?video_id=fBE7y6Uba5M&t=
vjVQa1PpcFPfHDFKYQ1s_RIHTM-GxADM8vFGLxxc_rs=&el=detailpage&ps=&fmt=34&asv
=2&noflv=1"
(b) GET "http://v8.nonxt7.c.YouTube.com/
videoplayback?ip=0.0.0.0&sparams=id%2Cexpire%2Cip%2Cipbits%2Citag%2Calgor
ithm%2Cburst%2Cfactor&fexp=904405&algorithm=throttle-factor&itag=34&ipbit
s=0&burst=40&sver=3&expire=1266310800&key=yt1&signature=66222E9350B9BB5AC
68297F12AC1DCB4C53AAFDE.55B33FFFA04EBF001AF39A4F316E657FC318E0E5&factor=1
.25&id=efa3a0434887fdc0&redirect_counter=1"
```

These two request formats do not have an explicit association or reference to the media object, and the URIs are not cache friendly. The association to the media object is provided using a combination of an **id** and **format** tag.

For case (a), this association is provided by the **video_id** and **fmt** query parameters.

For case (b), this association is provided by the **id** and **itag** query parameters.

Media Flow Controller uses a combination of these query parameters to generate an internal cache name for the media object. YouTube videos in Media Flow Controller are cached with a cache name format as:

```
yt_video_id_efa3a0434887fdc0_fmt_34
```

Media Flow Controller also supports random access using seek/scrub for YouTube videos. YouTube signals a seek point using a query parameter, **begin**, with units of milliseconds. Media Flow Controller translates this seek point to the correct position in the video file for both the FLV or MP4 container formats, and data that is delivered to the player is from the seek point onwards to the end of the file.

Configuring YouTube Video Caching

To configure YouTube video caching:

1. Configure a virtual player of **type youtube**:
   - The **cache-name** and **seek-config** configuration are required.
   - For **seek-config** configuration, the URI query is **begin** (seek length is not required).
   - The cache name configuration is **video-id** and **format-tag**.

   ```
   virtual-player youtube_player type youtube
     cache-name video-id query-string-parm "id" format-tag query-string- parm "itag"
     exit
   ```
2. Create a **namespace**, and set **match**, **origin-server**, and **domain** name values to filter YouTube video requests. The ***client_traffic_NIC*** for the **origin-server** must first be set with the **delivery protocol** command if this is for a transparent proxy deployment. See"Transparent Proxy Example Configuration—YouTube" on page 468 for details.

   ```
   namespace name
     match uri /videoplayback OR match uri /get_video
     origin-server http follow header host use-client-ip
     domain regex "^.*\.c\.youtube\.com|^.*\.googlevideo\.com"
   ```
3. Override the default cache age allowed for YouTube assets using **namespace origin-fetch** options to enable longer cache intervals. From **namespace** prefix mode set:

   ```
   delivery protocol http origin-fetch content-store media cache-age-threshold 300
   delivery protocol http origin-fetch cache-directive no-cache override
   delivery protocol http origin-fetch redirect-302 handle
   delivery protocol http origin-request host-header inherit incoming-req permit
   delivery protocol http origin-request x-forwarded-for disable
   ```
4. Allow Media Flow Controller to stop downloading an object after the client stops viewing it. If an object is partially cached, then on a second subscriber request, the remainder object is downloaded using a byte-range request. If the origin doesn't support byte-range requests, it sends the whole object and Media Flow Controller discards the part that has already been stored. From **namespace** prefix mode set:

   ```
   delivery protocol http origin-fetch cache-fill client-driven
   ```
5. Add the **virtual-player** you configured; activate the **namespace**, exit, and save your configuration. From **namespace** prefix mode set:

   ```
   virtual-player youtube_player
   ```

namespace status active
write memory

Example:

```
MFC (config) # virtual-player ytplayerA type youtube
MFC (config virtual-player ytplayerA) # cache-name video-id query-stringparm
id format-tag query-string-parm itag
MFC (config virtual-player ytplayerA) # seek-config query-string-parm begin
MFC (config virtual-player ytplayerA) # exit
MFC (config) # delivery protocol http interface client_traffic_NIC
MFC (config) # delivery protocol http transparent client_traffic_NIC enable
MFC (config) # namespace youtube
MFC (config namespace youtube) # delivery protocol http origin-fetch
cache-fill client-driven
MFC (config namespace youtube) # delivery protocol http origin-fetch
content-store media cache-age-threshold 300
MFC (config namespace youtube) # domain regex
"^.*\.c\.youtube\.com|^.*\.googlevideo\.com"
MFC (config namespace youtube) # match uri /videoplayback precedence 5
MFC (config namespace youtube) # origin-server http follow header host
use-client-ip
MFC (config namespace youtube) # virtual-player ytplayerA
MFC (config namespace youtube) # status active
MFC (config namespace youtube) # exit
MFC (config) # write memory
```

NOTE: For virtual player tuning, see "Transparent Proxy Cache Tuning Examples" on page 474.

Bit Rate Upgrade/Downgrade Support for YouTube Videos

Media Flow Controller offers the capability to switch an incoming request for a YouTube video to an already available lower or higher bit rate video in its media cache prior to client delivery. YouTube encodes videos in a variety of media formats, and for each format a set of different bit rates are provided. The YouTube client player specifically requests a particular format and bit rate each time a user accesses YouTube videos. YouTube player defaults to a fixed format and bit rate and allows the viewer to change this format to a higher or lower quality if needed (Upgrade/Downgrade).

The feature to upgrade and downgrade videos in the Media Flow Controller provides service providers with the following benefits:

- Save or optimize delivery bandwidth for YouTube content by switching all incoming video requests to a lower bit rate version of the same format.
- Provide a higher quality of service experience to premium customers for incoming video requests by switching them to a higher bit rate version of the same format.

By default, the YouTube Bit Rate Upgrade/Downgrade option is disabled in the YouTube virtual player.

- Understanding Video Formats Supported by YouTube on page 133
- Downgrading the YouTube Video Delivery Format on page 134

- Upgrading the YouTube Video Delivery Format Bit Rate on page 135
- Configuring Bit Rate Upgrade/Downgrade Support for YouTube Videos on page 136

Understanding Video Formats Supported by YouTube

Table 16 on page 133 defines the video formats supported by YouTube.

Table 16: YouTube Video Formats and Bit Rates

YouTube	Container Type	Video Codec	Video Bit Rate (kbps) *Approximate	Audio Codec	Audio Bit Rate (kbps) *Approximate
5	FLV	Sorenson H.263	250	MP3	64
34	FLV	H.264/AVC Main Profile	500	ACC	128
35	FLV	H.264/AVC Main Profile	800-1000	ACC	128
18	MP4	H.264/AVC Baseline Profile	500	ACC	96
22	MP4	H.264/AVC High Profile	2000-3000	ACC	152
37	MP4	H.264/AVC High Profile	3500-5000	ACC	152
43	WebM	VP8 (360p)	500	Vorbis	128
44	WebM	VP8 (480p)	600-1200	Vorbis	128
45	WebM	VP8 (720p)	2000 and above	Vorbis	192

NOTE: The bit rates documented in Table 16 on page 133 are approximate values observed in Media Flow Controller tests. YouTube generally encodes within these limits, however some files for formats might have bit rates above or below the rates documented in Table 16 on page 133.

NOTE: Table 16 on page 133 documents the formats and tags most often observed and used by YouTube clients. YouTube might introduce new formats or change the tags of existing ones, upon which this table may not serve as the most recent reference.

WARNING: The following caveats apply to the Upgrade/Downgrade feature supported in Media Flow Controller:

- Media Flow Controller only supports format upgrade and downgrade transition for MP4
- Cross format upgrade and downgrade is not supported, because it breaks Web browser behavior and FLV video formats.
- The WebM video format is not supported for upgrade and downgrade operation since it impacts seek operations.
- Upgrade and downgrade, in this release, is supported only if a lower or higher rate video is already present in the cache. If not, Media Flow Controller honors the original client request.

NOTE: Upgrade and downgrade requests are only supported for non-byte range queries.

Downgrading the YouTube Video Delivery Format

Table 17 on page 134 defines how Media Flow Controller downgrades the YouTube video delivery format.

Table 17: YouTube Video Delivery Formats for Downgrading

YouTube Video Format (Tag Values)	Container Type	Bit Rate (kbps) *Approximate
5	FLV	250
34	FLV	500
35	FLV	800-1000
18	MP4	500
22	MP4	2000-3000
37	MP4	3500-5000

The following behavior is expected from the cache, if you enable downgrade in the YouTube virtual player in Media Flow Controller. The final outcome is dependent on the

particular video format requested by the client player and the value configured in the Media Flow Controller CLI to clamp the bit rate (such as, limit-to rate).

Media Flow Controller checks which lowest bit rate video for a format is fully present in the cache. If this rate is above the limit-to rate, Media Flow Controller serves this format. If the lowest rate for this format is not available, Media Flow Controller checks for the next highest rate. If none succeeds, Media Flow Controller serves the original request.

For example:

- You set limit-to rate to 400 Kbps and Media Flow Controller receives a request for FLV format 35. If format 34 is present in cache (500 Kbps exceeds the lower clamp), Media Flow Controller downgrades to format 34. If format 34 is not present, Media Flow Controller serves the requested format 35.
- You set limit-to rate to 200 Kbps and Media Flow Controller receives a request for FLV format 35. If format 5 is present in cache (250 Kbps exceeds the lower clamp), Media Flow Controller downgrades to format 5. If format 5 is not present, Media Flow Controller checks for 34, and on failure, reverts to the requested format 35.

NOTE: **The downgrades stay within the respective format types. For FLV (5, 34, and 35) and for MP4 (18, 22, and 37).**

Upgrading the YouTube Video Delivery Format Bit Rate

Table 18 on page 135 defines how Media Flow Controller upgrades the YouTube video delivery format.

Table 18: YouTube Video Delivery Formats for Upgrading

YouTube Video Format (Tag Values)	Container Type	Bit Rate (kbps) *Approximate
5	FLV	250
34	FLV	500
35	FLV	800-1000
18	MP4	500
22	MP4	2000-3000
37	MP4	3500-5000

The following behavior is expected from the cache, if you enable the upgrade feature in the YouTube virtual player in Media Flow Controller. The final outcome is dependent on the particular video format requested by the client player and the value configured in the Media Flow Controller CLI to clamp the bit rate (such as, limit-to rate).

Media Flow Controller checks which highest bit rate video for this format is fully present in the cache. If this rate is below the limit-to rate, Media Flow Controller serves this format. If the highest rate for this format is not available, Media Flow Controller check for the next lowest rate. If none succeeds, Media Flow Controller serves the original request.

For example:

- You set limit-to rate to 6000 Kbps, and Media Flow Controller receives a request for MP4 format 18. If format 37 is present in cache (5000 Kbps is within the upper clamp), Media Flow Controller upgrades to format 37. If format 37 is not present, then Media Flow Controller checks if format 22 is available. If not, it serves the requested format 18.
- You set limit-to rate to 2000 Kbps and Media Flow Controller receives a request for MP4 format 18. If a higher rate MP4 format video matching format 22 is present that is less than 2000 Kbps, Media Flow Controller upgrades to that format. Otherwise, Media Flow Controller serves the origin request of format 18.

NOTE: **The upgrades stay within the respective format types. For FLV (5, 34, and 35) and for MP4 (18, 22, and 37).**

Configuring Bit Rate Upgrade/Downgrade Support for YouTube Videos

To configure bit rate upgrade and downgrade support for YouTube videos, use the following CLI commands or the YouTube virtual player:

```
[no] bandwidth-opt file-type {MP4 | FLV} switch-rate { higher | lower } limit-rate-to <kbps>
```

To enable YouTube bit rate downgrade support, use the following example configuration:

```
virtual player type youtube
bandwidth-opt file-type MP4 switch-rate higher limit-rate-to 3000
bandwidth-opt file-type FLV switch-rate lower limit-rate-to 500
bandwidth-opt file-type FLV switch-rate lower limit-rate-to 500
```

The preceding configuration switches or upgrades any incoming MP4 video file request to a rate that comes closest to 3000 kbps (not-exceeding) in the cache. Similarly, any incoming request for FLV format downgrades a file request to a rate that comes closest to 500 kbps.

Related Documentation

- Before You Configure Media Flow Controller on page 20
- Virtual Players Overview on page 117
- Virtual Player Features on page 120
- Configuring Rate-Control to Support Video Pacing for Client Players on page 124
- Virtual Player Features on page 120
- Media Flow Controller Virtual Players on page 16

Configuring the Smoothstream-Pub Virtual Player

Smooth Streaming Overview

Smooth Streaming is an HTTP-based adaptive streaming technology implemented by Microsoft. The media format defined by Microsoft for smooth streaming supports both storage and on-the-wire delivery, and is based on the ISO/IEC 14496-12 ISO Base Media File Format specification. Smooth Streaming technology requires content to be encoded at multiple bit rates that is then delivered to clients (for example, Microsoft Silverlight Player) as a series of small chunks or fragments. This allows the client player to dynamically switch between fragments of different bit rates or qualities, depending on network bandwidth and CPU state, allowing viewers to have the best possible experience.

The **type smoothstream-pub** virtual player supports publishing Smooth Streaming format ISMV files to fragmented format for the Silverlight player similar to IIS Windows Server.

Smooth Streaming is an HTTP-based adaptive streaming technology implemented by Microsoft. You would implement Smooth Streaming if a Microsoft Internet Information Server (IIS) is not deployed between the Media Flow Controller and your network. The media format defined by Microsoft for smooth streaming supports both storage and on-the-wire delivery, and is based on the ISO/IEC 14496-12 ISO Base Media File Format specification. Smooth Streaming technology requires content to be encoded at multiple bit rates which is then delivered to clients (such as, Microsoft Silverlight Player) as a series of small chunks or fragments. This allows the client player to dynamically switch between fragments of different bit rates or qualities depending on network bandwidth, CPU state, and so forth, allowing viewers to have the best experience possible.

Smooth Streaming Multiple Bit-Rate Assets

A typical multiple bit-rate media asset encoded according to the Smooth Streaming format is comprised of the following files:

- Media files containing video, audio (*.ismv or *.isma), or both—Denoted by the **ismv** and **isma** extensions. The format supports storage of either a single bit rate per file or can package all bit rates into a single file.
- Server manifest file (*.ism)—Denoted by the ***ism** extension. These are XML files which describe to the streaming server the relationship among media tracks, their bit rates, and files on disk, in the smooth streaming package.
- Client Manifest File (*.ismc)—Denoted by the *.ismc extension. This is typically the first file requested by a client and describes to the client the streams, codecs, encoded bit rates, and video resolutions that are available in this package.

Example: Smooth Streaming Workflow

A typical workflow between a player or client and the server is as follows:

1. The client requests from the server a client manifest (*.ismc) file. The manifest describes to the client which bit rates and resolutions are available, and includes a list of all the available chunks and either their start times or durations.
2. Thereafter the client requests fragments in the form of specific RESTful URLs, for example:

```
http://test.media.com/bunny.ism/QualityLevels(400000)/
  Fragments(video=134345672)
http://test.media.com/bunny.ism/QualityLevels(64000)/
  Fragments(audio=123452356)
```

The values passed in the URL represent bit rate (**400000**) and the fragment start offset (**134345672**).

With the **virtual-player type smoothstream-pub** configured, Media Flow Controller supports the delivery of Microsoft's Smooth Streaming media assets. This virtual player supports on-demand, file-based media only; it does not support live streaming. It can read the RESTful URLs and can fragment in real time by checking the specified quality level or fragment offset in the URL. Internally, the virtual player reads the server manifest (*.ism) and locates the physical *.ismv or *.isma file. It can then parse the .ismv or .isma files according to the Smooth Streaming specification and locate the fragment box ('moof' + 'mdat') that corresponds to the requested start time offset. Media Flow Controller can extract this fragment box and send it to the client as a standalone file with the correct content type (video/mp4, audio/mp4, text/xml, and so on).

The fragmentation functionality can be deployed in multiple scenarios; for example, a single Media Flow Controller at the origin to perform the fragmentation and multiple Media Flow Controller edge caches caching and delivering the fragments. Or, the fragmentation functionality can be deployed at each edge location.

Configuring Smooth Streaming Caching and Delivery (CLI)

To configure Smooth Streaming caching and delivery:

1. Create the Smooth Streaming virtual player.

   virtual player *name* type smoothstream-pub

2. In prefix mode, define the **quality-tag** identifier to describe to Media Flow Controller the bit rate of the requested media segment. The default is **QualityLevels** (case sensitive).

   quality-tag *string*

3. In prefix mode, define the **fragment-tag** identifier whose value describes to Media Flow Controller the timestamp of the requested media segment. The default is **Fragments** (case sensitive).

   fragment-tag *string*

For example, in the **quality-tag** identifier and **fragment-tag** identifier would be the parameters specified in parentheses in the URL below:

For example, in the **quality-tag** identifier and **fragment-tag** identifier would be the parameters specified in parentheses in the URL below:

Here the **quality-tag** is configured as **QualityLevels**, and the **fragment-tag** is configured as **Fragments**.

Configuring the Flashstream-Pub Player

FlashStreaming Overview

Adaptive HTTP streaming support to the Flash Platform commonly referred to as HDS (HTTP Dynamic Streaming) or Zeri was introduced by Adobe recently.

The **flashstream-pub** virtual player supports publishing Adobe Zeri/HDS assets to fragmented format for the Flash player, similar to the origin module functionality provided by Adobe/FMS.

HTTP adaptive streaming support to the Flash Platform, commonly referred to as HDS (HTTP Dynamic Streaming), was introduced by Adobe recently. The technology requires media assets to be packaged or pre-published into fragments that Flash player clients can access without the need to download an entire file.

To support HDS in Flash, several components in the publishing and delivery workflow must be addressed. Media assets must be packaged in the Zeri format. Delivery servers or caches must understand the HDS fragment requests sent by a player. The servers/caches also must perform in-line fragmentation to deliver the right fragment when configured as origin-servers or caches.

NOTE: To publish assets according to the Zeri format, Adobe provides a utility called the File Packager, which translates multiple bit-rate media assets into segments as F4F files along with the manifest file (F4M) and an index file (F4X). The File Packager is available from adobe.com and is installed with Adobe® Flash® Media Server in the rootinstall/tools/f4fpackager folder.

Delivery servers, particularly those that front-end origin stores or libraries hosting Zeri formatted media must be capable of performing on-demand fragmentation of the assets into the respective Zeri fragments (F4F). In Media Flow Controller, a new **virtual-player type**, **flashstream-pub**, can serve on-demand files created by the File Packager (Adobe). This virtual player supports on-demand, file-based media only. It does not support live streaming.

The **flashstream-pub** virtual player can process Flash Fragment File Format (F4F), Flash Manifest File Format (F4M), and Flash Index File Format (F4X) requests.

Example: FlashStreaming Workflow

In this example, the a sample player that supports HDS is the OSMF player, built using the Open Source Media Framework (OSMF) which runs in Flash Player 10.1 (osmf.org).

The workflow is as follows:

1. The player first requests for a manifest file (F4M file).

   http://*server-name/path-to-asset/manifest-file-name*.f4m

2. The player receives the manifest file, parses the bootstrap information, and then creates a request by mapping a video time-code to a Segment#-Fragment# combination.
3. The player sends this request to Media Flow Controller, requesting for a specific fragment from an F4F file.

   http://*server-name/path-to-asset/*foo1000Seg1-Frag4

4. Media Flow Controller receives this request and the virtual player flashstream-pub parses the request to infer the video asset name (foo1000), segment number (1), and fragment number (4).
5. Media Flow Controller fetches the corresponding index file (F4X: foo1000Seg1.f4x) for the asset segment (foo1000Seg1) and finds the offset to the fragment number 4 in the segment file (F4F: foo1000Seg1.f4f).
6. Media Flow Controller then serves only the required bits from the obtained byte offset which corresponds to the fragment number 4 for this asset to the player.
7. The Flash player receives this fragment and starts playback.
8. Once the asset is fragmented, the delivery or caching systems further down the pipeline do not need a virtual player as the data is now pre-chunked.

Configuring FlashStreaming Caching and Delivery (CLI)

To configure FlashStreaming caching and delivery:

1. Create the FlashStream virtual player.

   virtual player *name* type flashstream-pub

2. Define the **segment-tag** identifier that describes to Media Flow Controller the string to search while parsing a Zeri HDS request. The number immediately following this tag in the request denotes the segment number.

   segment-tag *string*

3. Define the **fragment-tag** identifier that describes to Media Flow Controller the string to search while parsing a Zeri HDS request. The number immediately following this tag in the request denotes the fragment number.

   fragment-tag *string*

4. Define the **seg-frag-delimiter** identifier that describes to Media Flow Controller the separator to search while parsing a Zeri HDS request which acts as a delimiter to the segment and fragment tags.

   seg-frag-delimiter *string*

For example; a typical HDS request for a fragment is of the form:

`http://server-name/path-to-asset/foo1000Seg1-Frag4`

Here the **segment-tag** is configured as **Seg**, the **fragment-tag** as **Frag**, and the **seg-frag-delimiter** as **-** (dash).

Related Documentation

- Before You Configure Media Flow Controller on page 20
- Virtual Players Overview on page 117
- Virtual Player Features on page 120
- Configuring Rate-Control to Support Video Pacing for Client Players on page 124
- Virtual Player Features on page 120
- Media Flow Controller Virtual Players on page 16

Configuring the Virtual Player Type yahoo

NOTE: You must install a license to configure the Yahoo virtual player.

The **yahoo** virtual player includes **connection max-bandwidth**, and **seek** options (see "Configuring the Generic Virtual Player" on page 127 for details), as well as these special options:

- **health-probe**—Configures an external server to do health checks by making Media Flow Controller fetch data from origin and play it to the server initiating the health check. The signal that a given HTTP request is for a health probe is the **health-probe query-string-parm name**. If that **name** value matches the subsequent ***string*** value, the GET request is treated as a health probe. When servicing health probes, Media Flow Controller does not cache the data into disk or in buffer. Use **virtual player *name* type 3 no health-probe** to disable.
- **req-auth**—Computes an MD5 hash of query string parameters representing **stream-id**, **auth-id**, a configured **shared-secret**, and **time-interval**; and matches the computed value with the specified **match query-string-parm *string***. The HTTP GET proceeds if the computed MD5 hash matches; if there is no match, the session is rejected. Use **virtual player *name* type 3 no req-auth** to disable.

Related Documentation

- Before You Configure Media Flow Controller on page 20
- Virtual Players Overview on page 117
- Virtual Player Features on page 120

- Configuring Rate-Control to Support Video Pacing for Client Players on page 124
- Virtual Player Features on page 120
- Media Flow Controller Virtual Players on page 16

CHAPTER 5

Configuring Media Fetch and Pre-Staging (CLI)

Media Fetch Overview

You can publish content into the Media Flow Controller cache file system in multiple ways, such as on-demand fetch from an origin server, and scheduled crawls. The origin server is the repository of all the media assets. An origin server is typically owned by a content owner, content distributor, or Content Delivery Network (CDN) service provider. The methods to publish content into the Media Flow Controller cache are as follows:

- On-demand fetch—You can configure the Media Flow Controller to perform an on-demand fetch of the content from the origin server, if the content in not present in the cache (that is, RAM or the disks).
- Scheduled preload—A content provider can configure the Media Flow Controller to preload a list of objects to the cache at a scheduled time. The content providers give a list of URLs to be preloaded in a text file, and the system pulls the preload URL list file from a remote server using SCP. Finally, the Media Flow Controller begins preloading the objects at the scheduled time using HTTP.
- Scheduled crawl—You can configure the Media Flow Controller to crawl content from an origin server at scheduled intervals. The Content Ingest Manager crawls periodically to ensure that the content in the cache is in sync with the content at the origin server.

You use the following protocols to publish content from the origin server to the cache in the Media Flow Controller:

- HTTP
- HTTPS

- RSTP
- NFS
- FTP

Configuring NFS Fetch for Images (CLI)

Configuring Media Flow Controller to fetch all JPEG image files located in a particular directory on an NFS origin server requires another **namespace** configuration.

In this example, all requests for files at **www.example.com/jpgImages/** that incur a cache miss (necessitating an origin fetch) are fetched from the specified origin using NFS; the configuration could look like this:

```
MFC (config) # namespace exampleJpegs
MFC (config namespace exampleJpegs) # domain www.example.com
MFC (config namespace exampleJpegs) # match uri /jpgImages/
MFC (config namespace exampleJpegs) # origin-server nfs exampleOS.com: /
home/jpgImages
MFC (config namespace exampleJpegs) # status active
MFC (config namespace exampleJpegs) # exit
MFC (config) #
```

The **match uri-prefix** of **/jpgImages** tells Media Flow Controller to use the defined namespace rules for incoming requests containing /jpgImages in the URL. NFS will fetch the requested file from the specified origin server on the default port (the default NFS port is 2049).

Just as with the initial namespace configuration, repeat the namespace configurations with the **domain** specified without the "www" prefix, and again specified with the IP address, if requests can come in those ways. Test the configuration with **ping**.

TIP: If unsure what port you are using for origin-server, use standard Linux shell commands to figure out the port, and then configure it along with the origin server, if not the default.

Related Documentation

- Configuring RTSP Fetch for Videos (CLI) on page 145
- Configuring HTTP Fetch for Videos (CLI) on page 144
- Media Flow Controller Delivery Methods on page 9

Configuring HTTP Fetch for Videos (CLI)

Configuring Media Flow Controller to fetch all video files, located in a particular directory on the origin server, using HTTP, requires another namespace configuration. In this example, all requests for videos at **www.example.com/videos/** that incur a cache miss (necessitating an origin fetch) are fetched from the specified origin using HTTP.

For our example company, **www.example.com**, the configuration could look like this:

```
MFC (config) # namespace exampleVideos
MFC (config namespace exampleVideos) # domain www.example.com
MFC (config namespace exampleVideos) # match uri /videos/
MFC (config namespace exampleVideos) # origin-server http
  www.exampleOS.com 80
MFC (config namespace exampleVideos) # status active
MFC (config namespace exampleVideos) # exit
MFC (config) #
```

The **match uri-prefix** of **/videos** tells Media Flow Controller to use these namespace rules for incoming requests containing /videos in the URL. HTTP is used to fetch the requested file from the specified origin server on port 80 (the HTTP default). Just as with the initial namespace configuration, repeat the namespace configurations with the **domain** specified without the " www" prefix, and again specified with the IP address, if requests may come in those ways.

TIP: If unsure what port you are using for **origin-server**, use standard Linux shell commands to figure out the port, and then configure it along with the origin server, if not the default.

Related Documentation

- Configuring RTSP Fetch for Videos (CLI) on page 145
- Configuring NFS Fetch for Images (CLI) on page 144
- Media Flow Controller Delivery Methods on page 9

Configuring RTSP Fetch for Videos (CLI)

Configuring Media Flow Controller to fetch all video files, located in a particular directory on the origin server, using RTSP, requires another namespace configuration. In this example, all requests for videos at **www.example.com/videos/** that incur a cache miss (necessitating an origin fetch) are fetched from the specified origin using RTSP.

For our example company, **www.example.com**, the configuration could look like this:

```
MFC (config) # namespace exampleVideos
MFC (config namespace exampleVideos) # domain www.example.com
MFC (config namespace exampleVideos) # match uri /videos/
MFC (config namespace exampleVideos) # origin-server rtsp
  www.exampleOS.com 554
MFC (config namespace exampleVideos) # status active
MFC (config namespace exampleVideos) # exit
MFC (config) #
```

The **match uri-prefix** of **/videos** tells Media Flow Controller to use these namespace rules for incoming requests containing /videos in the URL. RTSP is used to fetch the requested file from the specified origin server on port 554 (the RTSP default).

Just as with the initial namespace configuration, repeat the namespace configurations with the **domain** specified without the "www" prefix, and again specified with the IP address, if requests may come in those ways.

TIP: If unsure what port you are using for origin-server, use standard Linux shell commands to figure out the port, and then configure it along with the origin server, if not the default.

Related Documentation

- Configuring NFS Fetch for Images (CLI) on page 144
- Configuring HTTP Fetch for Videos (CLI) on page 144
- Media Flow Controller Delivery Methods on page 9

Pre-Fetching Content to the Cache (CLI)

To help prevent flash crowds, you can pre-load content to Media Flow Controller in advance. HTTP pre-fetch is done by copying a file of URLs to Media Flow Controller. Those URLs must be **http://**. That URL list is copied using SCP; the objects are fetched using HTTP.

To pre-load content:

1. Create a text file containing a list of URIs to preload. Put each URI in the text file on a separate line using this syntax:

   http://{*domain*} [:*port*]/{URI}

2. Pre-load the text file, do not include a suffix. Optionally, trigger the **pre-fetch** at a particular **fetch-time** (*hh:mm:ss*) in the same day; or, specify a **fetch-time** for another day (*yyyy/mm/dd*) and time. All time formats are UTC.

   pre-fetch *SCP:URI_list_text_file* (fetch-time [*time*] [*date*])

   Media Flow Controller parses the uploaded file for URIs and triggers a loopback "curl" request to all the URIs, one by one. This causes a cache-miss for each URI and an origin-fetch and the caching of the URIs.

3. Verify that the URIs are pre-loaded:

   namespace *name* object list

4. To delete the scheduled jobs:

   no pre-fetch *filename*

5. To list the scheduled prefetch jobs:

   show pre-fetch list

   Examples:

   pre-fetch scp://root:passwd@10.19.172.91/prefetch/prefetch.txt
   pre-fetch scp://root:passwd@10.19.172.91/prefetch/prefetch.txt fetch-time 12:30
   pre-fetch scp://root:passwd@10.19.172.91/prefetch/prefetch.txt fetch-time 12:30 2010/10/10

HTTPS Origin Fetch from Origin Servers

HTTPS Support and SSL Client Delivery Overview

Media Flow Controller supports Secure Sockets Layer (SSL) termination used by Content Delivery Networks (CDNs) and content publishers for Hyper Text Transfer Protocol over SSL (HTTPS) to users. It fetches media from origin servers using HTTP/NFS (or FTP ingest) and delivers using HTTPS. HTTPS is the standard encrypted communication mechanism on the Web.

SSL allows clients to connect to Media Flow Controller securely. SSL uses public key encryption to authenticate the server to the client.

HTTPS support in Media Flow Controller provides the following benefits:

- Consolidates the delivery of multiple protocols, including HTTP, HTTPS, RTSP, and RTMP from the same server, thereby reducing the number of servers used for media delivery.
- Eliminates the SSL processing overhead from origin servers.

Media Flow Controller performs an SSL handshake with the client and establishes a unique session key used for encryption. Media Flow Controller shares the public key in the SSL certificate with clients which is used for deciphering the encrypted data. It allows CDNs and content publishers to upload certificates, import certificates from a remote server, and delete certificates.

SSL support is a separately licensed Media Flow Controller feature. You must install the Media Flow Controller SSL license before you can configure SSL. Media Flow Controller uses SSL versions 3.0 and 2.0 and Transport Layer Security (TLS) version 1.0 standards for HTTPS certificate authentication.

Media Flow Controller supports Server Name Identification (SNI) to send the correct certificate to the user during TLS negotiation based on the virtual host to which the request maps. It provides an option to upload a certificate and use that for one or more namespaces belonging to a CDN customer. This is achieved by matching the received client request server or domain name with the virtual hostname.

Media Flow Controller supports ciphers (see "Configuring the SSL Key File" on page 151 and cipher suites (see "Configuring the Virtual-Host Cipher String" on page 161) provided by OpenSSL for SSL/TLS data encryption.

To configure HTTPS and SSL client delivery, perform these tasks in the order presented:

- Installing the Media Flow Controller SSL License on page 148

NOTE: **SSL support is a separately licensed feature. You must install an SSL license before you can configure SSL parameters.**

- Configuring the HTTPS Delivery Protocol Interface on page 149
- Configuring HTTPS Protocol Listen Port Numbers on page 150
- Configuring the SSL Key File on page 151
- Configuring the SSL CSR File on page 154
- Configuring an SSL Certificate File on page 156
- Binding an SSL Certificate and Key to a Virtual Host on page 160
- Configuring the Virtual-Host Cipher String on page 161
- Enabling HTTP and HTTPS Client Delivery per Namespace on page 163
- Mapping HTTPS Namespaces and Virtual Hosts on page 165
- Configuring HTTPS Fetch from Origin Servers on page 166
- Verifying HTTPS Client Delivery on page 167

Installing the Media Flow Controller SSL License

Media Flow Controller supports the toolkit for SSL/TLS. SSL support is a separately licensed Media Flow Controller feature. Juniper Networks provides the license to customers to enable SSL client delivery. You must install the Media Flow Controller SSL license before you can configure SSL.

To install the Media Flow Controller SSL license:

1. Install the SSL license.

   # license install *license key*.

   # license install mfc_ssl_license

2. View the license.

```
# show licenses

    License 3:
    LK2-SSL-44HT-1L68-RK4E-9P8M-1JTC-T46C-VJTH-226L-U2TG-RQ6G-V2TE-9H8C-R
    36H-9G6W-242C-T381-GT1P-6VMK-14RB-V8G4-X5XP-UJNF-2CM8
      Feature:     SSL
      Valid:       yes
      Tied to host ID:  423297EC-3D39-DB56-C848-91C23E07DA3C (ok)
      Active:      yes
```

To delete the Media Flow Controller SSL license:

1. Delete the license.

   # license delete mfc_ssl_license

Related Documentation

- HTTPS Support and SSL Client Delivery Overview on page 147
- Configuring the SSL Key File on page 151
- Configuring the SSL CSR File on page 154
- Configuring an SSL Certificate File on page 156
- Binding an SSL Certificate and Key to a Virtual Host on page 160
- Configuring the Virtual-Host Cipher String on page 161

Configuring the HTTPS Delivery Protocol Interface

Enabling HTTPS delivery under an interface, (such as eth1, eth2) requires that you restart the Media Flow Controller SSL service. You can use any interface for HTTPS delivery. However, it is recommended that you reserve eth0 for management.

To configure the HTTPS delivery interface:

1. Enter the CLI configure mode.

   > **enable**

   # configure terminal

2. Configure one or more HTTPS protocol interface.

   You can configure any interface. You can set up to 10 interfaces.

   (config) # **delivery protocol https listen interface *interface-name***

   For example, enter **delivery protocol https interface eth0.**

   WARNING: **Restart the mod-ssl service for the reset to take effect. See Step 3.**

3. Restart the SSL service.

   # service restart mod-ssl

4. View the configured HTTPS protocol interface.

   # **show delivery protocol https**

   You see the following HTTPS protocol delivery information:

   ```
   Delivery Protocol : HTTPS
   Server Ports:
     443
   Interfaces:
     Interface: eth0
   ```

   Use the **no delivery protocol https interface <*interface-name*>** command to clear the HTTPS listen interface.

   WARNING: **Restart the mod-ssl service for the reset to take effect. See Step 3.**

5. Exit the CLI enable mode.

   # **exit**

Related Documentation

- HTTPS Support and SSL Client Delivery Overview on page 147
- Configuring HTTPS Protocol Listen Port Numbers on page 150
- Verifying HTTPS Client Delivery on page 167

Configuring HTTPS Protocol Listen Port Numbers

You must enable the port(s) on which Media Flow Controller listens for incoming HTTPS requests from clients. You can configure one or more HTTPS protocol listen port numbers. By default, SSL listens on port 443. When you configure the port numbers, you must restart the Media Flow Controller SSL service for the configuration to take effect.

To configure the HTTPS listen port number:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Configure one or more HTTPS protocol listen port numbers.

   (Config) # **delivery protocol https listen port <*port number* > <*port number*> <*port number*>**

   For example, enter **delivery protocol https listen port 234 345 443**. You can configure up to 64 ports.

   WARNING: **Restart the mod-ssl service for the HTTPS listen port reset to take effect.**

You can remove any port number using the **no delivery protocol https list port** <*port number*> CLI command.

3. Restart the SSL service.

   # **service restart mod-ssl**

4. View the listen port number(s) that you configured.

   # **show delivery protocol https**

   ```
   Delivery Protocol : HTTPS
   Server Ports:
     443
   Interfaces:
     Interface: bnd0
   ```

5. Exit the CLI enable mode.

   # exit

Related Documentation

- HTTPS Support and SSL Client Delivery Overview on page 147
- Configuring the HTTPS Delivery Protocol Interface on page 149
- Verifying HTTPS Client Delivery on page 167

Configuring the SSL Key File

An RSA private key file is a digital file used to decrypt incoming messages sent to Media Flow Controller. The key has a public component that you distribute (using your Certificate file) allowing people to encrypt messages to you.

You can use a ***.pem** key file to configure the SSL certificate. Certificates might come with a ***.crt** extension; however, Media Flow Controller supports only the ***.pem** extension. Many ***.crt** certificates have a PEM format, and you need only change the filename to ***.crt**. To check whether the ***.crt** certificate is of PEM format, open the certificate in a text editor and look for the **–BEGIN CERTIFICATE–** and **–END CERTIFICATE–** statements.

A cipher (or cypher) is an algorithm for encryption. A key varies the encryption. You select a key before using cipher encryption information. Without the key, it is impossible to decrypt cypher text into readable plain text.

You specify the key length when you create the SSL key file.

A passphrase is required to create the SSL key file.

You need the key file and passphrase of the key file when creating an SSL certificate.

Before You Begin:

- Ensure that you installed the SSL license. See “Installing the Media Flow Controller SSL License” on page 148.
- Ensure that you configured the HTTPS protocol interface. See “Configuring the HTTPS Delivery Protocol Interface” on page 149.

- Ensure that you configured the HTTPS protocol listen port numbers. See "Configuring HTTPS Protocol Listen Port Numbers" on page 150

This topic includes the following sections:

Creating an SSL Key File

To create an SSL key file:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Configure the SSL key CLI parameters.

   (config) # **ssl key <*name*> create cipher (des | des3 | aes128 | aes256) length <*value*> <*passphrase*>**

   The SSL key includes the following configurable parameters:

   - ***name***—The name of the RSA private ***.pem** key file.
   - **cipher encryption method**—Media Flow Controller supports the following symmetric key algorithms:
     - **DES**—Data Encryption Standard with key size of 56 bits.
     - **3DES**—Triple Data Encryption Standard applies the DES cipher algorithm times to each data block, and is therefore is more secure than DES. It is a preferred cipher for high security HTTPS transactions.
     - **AES128**—Advanced Encryption Standard in 128-bits-key size is a specification for the encryption of electronic data.
     - **AES 256**—Advanced Encryption Standard in 256-bits-key size is a specification for the encryption of electronic data.
   - ***value***—Media Flow Controller supports the following key lengths: 40, 56, 128, 168, 256, and 512 bits.
   - ***passphrase***—The passphrase is used while creating the passphrase protected key file. Use a nonsense phrase that has a different structure and words that don't quite logically connect. Add several random characters to make it impossible to guess by any means other than brute force.

   For example, enter **ssl key abc.pem create cipher aes128 length 512 abc123**.

   Media Flow Controller displays the following RSA private key output:

```
Generating RSA private key, 512 bit long modulus ......
++++++++++++ ..................
++++++++++++
e is 65537 (0x10001)
```

3. Exit enable CLI mode.

   # **exit**

Importing an SSL Key File

You can Import SSL ***.pem** key files to Media Flow Controller.

To import an SSL key file:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Configure the SSL key import SCP path.

   (Config) # **ssl key <*name*> import <*scp path*> <*passphrase*>**

   For example, enter **ssl key abc.pem import scp://host@10.157.42.57/home/username/abc.pem abc123**.

3. Exit enable CLI mode.

   # **exit**

Exporting an SSL Key File

You can export a ***.pem** SSL key file from the Media Flow Controller to an SCP path.

To export an SSL key file:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Export the SSL key to a location using SCP.

   (config) # **ssl key <*name*> export <*scp path*>**

   For example, enter **ssl key abc.pem export scp://host@10.157.42.57/home/username/**.

3. Exit enable CLI mode.

   # **exit**

Removing an SSL Key File

To remove an SSL key file:

1. Enter the CLI configure mode.

> **enable**

 # configure terminal

2. Remove an SSL key file.

   ssl key <*name*> remove

   For example, enter **ssl key abc.pem remove**.

3. Exit enable CLI mode.

   # exit

Related Documentation

- HTTPS Support and SSL Client Delivery Overview on page 147
- Installing the Media Flow Controller SSL License on page 148
- Configuring the SSL CSR File on page 154
- Configuring an SSL Certificate File on page 156

Configuring the SSL CSR File

The SSL Certificate Signing Request (CSR) allows you to create a self-signed certificate. The CSR is the digital file that contains your public key and your name. You send the CSR to a Certifying Authority (CA), that converts it into a real Certificate, by signing it.

This topic includes the following sections:

Creating the CSR File

To create an CSR file:

1. Enter the CLI configure mode.

   > **enable**

   # configure terminal

2. Configure the CSR file CLI parameters.

   (config) # **ssl csr <*name*> create input-key <*key-file name*> <*passphrase*>**

   For example, enter **ssl csr abc_csr.pem create input-key abc.pem abc123**

3. Answer the CSR questions.

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
```

```
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:US
State or Province Name (full name) [Berkshire]:California
Locality Name (eg, city) [Newbury]:Sunnyvale
Organization Name (eg, company) [My Company Ltd]:Juniper Networks
Organizational Unit Name (eg, section) []:CMBU
Common Name (eg, your name or your server's hostname) []:server01
Email Address []:myemaill@juniper.net

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:pwxyz123
An optional company name []:juniper
```

4. Exit enable CLI mode.

   # **exit**

Importing the CSR File

You can import SSL ***.pem** CSR files to the Media Flow Controller.

To import a CSR file:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Import a CSR file to Media Flow Controller.

   (configure) # **ssl csr <*name*> import <*scp path*>**

   For example, enter **ssl csr abc_csr.pem import scp://host@10.157.42.57/home/username/**.

3. Exit enable CLI mode.

   # **exit**

Exporting the CSR File

You can export an SSL CSR file from Media Flow Controller to an external location using SCP.

To export the CSR file:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Export the CSR file.

   (configure) # **ssl csr <*name*> export <*scp path*>**

For example, enter **ssl csr abc_csr.pem export scp://host@10.157.42.57/home/username/**.

3. Exit enable CLI mode.

   # **exit**

Removing the CSR File

To remove an SSL CSR file:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Remove the CSR file.

   (config) # **ssl csr <*name*> remove**

   For example, enter **ssl csr abc_csr.pem remove**.

3. Exit enable CLI mode.

   # **exit**

Related Documentation

- HTTPS Support and SSL Client Delivery Overview on page 147
- Installing the Media Flow Controller SSL License on page 148
- Configuring the SSL Key File on page 151
- Configuring an SSL Certificate File on page 156
- Binding an SSL Certificate and Key to a Virtual Host on page 160
- Configuring the Virtual-Host Cipher String on page 161

Configuring an SSL Certificate File

An SSL certificate, contains information about the owner of that certificate, including e-mail address, owner's name, certificate usage, duration of validity, resource location or Distinguished Name (DN) which includes the Common Name (CN) (web site address or e-mail address depending of the usage) and the certificate ID of the person who certifies (signs) this information. It contains also the public key and finally a hash to ensure that the certificate has not been tampered with. Browsers that know the Certificate Authority (CA) can verify the signature on that certificate, thereby obtaining your RSA public key. That enables them to send messages which only you can decrypt.

Back up the SSL certificate in a secure location.

This topic includes the following sections:

- Creating the SSL Certificate File on page 157
- Importing an SSL Certificate File on page 158

- Exporting an SSL Certificate File on page 159
- Removing the SSL Certificate File on page 159

Creating the SSL Certificate File

To create the SSL certificate file:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Configure the SSL certificate CLI parameters.

   (config) # **ssl certificate <*name*> create validity <*days*> input-key <*key-filename*> <*passphrase*>**

   For example, enter **ssl certificate abc_cert.pem create validity 365 input-key abc.pem abc123**.

   The SSL certificate includes the following configurable parameters:

   - ***name***—The SSL certificate name.
   - ***days***—The number of days the SSL certificates is valid.
   - ***filename***—The SSL key filename that you created when you created the SSL key file.
   - ***passphrase***—The SSL certificate password phrase used when you created the SSL key file.

   Media Flow Controller displays the following output:

   ```
   You are about to be asked to enter information that will be incorporated
   into your certificate request.
   What you are about to enter is what is called a Distinguished Name or a DN.
   There are quite a few fields but you can leave some blank
   For some fields there will be a default value,
   If you enter '.', the field will be left blank.
   -----
   Country Name (2 letter code) [GB]:CA
   State or Province Name (full name) [Berkshire]:California
   Locality Name (eg, city) [Newbury]:Sunnyvale
   Organization Name (eg, company) [My Company Ltd]:Juniper Networks
   Organizational Unit Name (eg, section) []:CMBU
   Common Name (eg, your name or your server's hostname) []:host02
   Email Address []:abc@xyz.com
   ```

3. View the SSL certificate.

   # **show ssl certificate <*filename*>**

   For example, enter **show ssl certificate abc_cert.pem**.

   ```
   show ssl certificate abc_cert.pem
   Certificate:
     Data:
   ```

```
    Version: 3 (0x2)
    Serial Number:
      db:bf:a8:10:1d:1c:3d:af
    Signature Algorithm: sha1WithRSAEncryption
    Issuer: C=CA, ST=Sunnyvale, L=Sunnyvale, O=Juniper Networks, OU=CMBU, CN
=host02/emailAddress=abc@xyz.com
    Validity
      Not Before: Jan 27 03:40:44 2012 GMT
      Not After : Jan 26 03:40:44 2013 GMT
    Subject: C=CA, ST=Sunnyvale, L=Sunnyvale, O=Juniper Networks, OU=CMBU, C
N=host02/emailAddress=abc@xyz.net
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (512 bit)
        Modulus (512 bit):
          00:b4:87:63:ae:4a:16:58:19:d1:88:23:bd:6b:ee:
          82:4f:20:54:7d:b5:4b:29:96:58:f5:cc:4e:15:97:
          d4:a6:25:49:8d:6c:65:fc:78:ba:39:ab:91:fb:3d:
          54:2e:36:86:e6:e4:90:e6:1e:5e:a1:ac:7a:ad:ec:
          b6:86:5e:9e:0f
        Exponent: 65537 (0x10001)
    X509v3 extensions:
      X509v3 Subject Key Identifier:
        8E:5F:61:58:DD:56:8D:BB:8E:C3:2B:87:4D:2B:18:37:86:C5:CD:03
      X509v3 Authority Key Identifier:
        keyid:8E:5F:61:58:DD:56:8D:BB:8E:C3:2B:87:4D:2B:18:37:86:C5:CD:03
        DirName:/C=CA/ST=Sunnyvale/L=Sunnyvale/O=Juniper Networks/OU=CMB
U/CN=host02/emailAddress=abc@xyz.comt
        serial:DB:BF:A8:10:1D:1C:3D:AF

      X509v3 Basic Constraints:
        CA:TRUE
  Signature Algorithm: sha1WithRSAEncryption
    89:73:38:97:b8:4f:d9:84:9b:74:79:d1:55:57:df:47:62:29:
    a2:e2:9d:12:3d:f5:9d:70:2f:ec:2e:a5:34:7c:3e:40:8e:48:
    3b:89:cc:13:22:f3:2c:00:f7:43:5b:60:87:eb:2b:5b:ff:24:
    e5:d1:03:00:c4:bb:ff:f6:e3:89
```

4. Exit enable CLI mode.

   # **exit**

Importing an SSL Certificate File

You can upload a SSL certificate ***.pem** file and secure copy it from that location to Media Flow Controller.

You can upload multiple SSL certificates. Media Flow Controller supports Server Name Identification (SNI) to send the correct certificate to the user during TLS negotiations based on the virtual host to which the request maps.

To import an SSL certificate file:

1. Enter the CLI configure mode.

   > **enable**

configure terminal

2. Configure the SSL certificate CLI parameters.

   (config) # **ssl certificate <*name*> import <*scp path*>**

   For example, enter **ssl certificate abc_cert.pem import scp://host@10.157.42.57/home/username/abc_cert.pem**

   ```
   Password (if required): *******
   100.0%
   ##################################################################################
   Import Successful
   ```

3. Exit enable CLI mode.

   # **exit**

Exporting an SSL Certificate File

To export the SSL certificate file:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Export the SSL certificate file.

   (config) # **ssl certificate <*name*> export <*scp path*>**

   For example, enter **ssl certificate abc_cert.pem export scp://host@10.157.42.57/home/username/abc_cert.pem**.

3. Exit enable CLI mode.

   # **exit**

Removing the SSL Certificate File

To remove the SSL certificate file:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Remove the SSL certificate file.

   (config) # **ssl certificate <*name*> remove**

   For example, enter **ssl certificate abc_cert.pem remove**

3. Exit enable CLI mode.

   # **exit**

Related Documentation

- HTTPS Support and SSL Client Delivery Overview on page 147
- Installing the Media Flow Controller SSL License on page 148
- Configuring the SSL Key File on page 151
- Configuring an SSL Certificate File on page 156
- Binding an SSL Certificate and Key to a Virtual Host on page 160
- Configuring the Virtual-Host Cipher String on page 161

Binding an SSL Certificate and Key to a Virtual Host

Transport Layer Security (TLS) extensions allow clients to include a Server Name Indication extension (SNI) in the extended ClientHello message. This extension hints to the server which name the client wishes to connect to, so the server can select the appropriate certificate to send to the client.

SSL context for each virtual domain is initialized during Media Flow Controller start up based on the CLI configuration. If the incoming client request does not map to any of the virtual-domain settings, Media Flow Controller uses the certificate, key, and ciphers corresponding to the wildcard virtual host.

Media Flow Controller supports 1000 concurrent sessions with SSL delivery throughput limited to 300 Mbps.

When you bind a virtual host to a SSL certificate, you need the following:

- Virtual host name or IP address. Currently Media Flow Controller supports configuring 256 virtual-hosts.
- Wildcard (used only in combination with the virtual host to match all IP addresses) or a fully qualified domain name for the IP address of the virtual host.
- SSL certificate name.
- SSL key name.

To bind an SSL certificate and key file to a virtual host:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Enter a virtual host wildcard or domain name and bind it to an SSL certificate name and key name

   (config) # **virtual-host [wildcard|*virtual-domain-name*] certificate bind <*cert-name*> key bind <*key-name*>**

   For example, enter **virtual-host www.juniper.net certificate bind abc_cert.pem key bind abc.pem**.

- **wildcard**—Binds the default certificate and key pair used when a client request does not match any of the configured virtual-hosts.
- *virtual-domain-name*—The IP address or domain name of the virtual host
- *cert-name*—The filename of the SSL certificate you created
- *key-name*—The filename of the SSL key you created

You can configure up to 256 virtual hosts.

For example,

3. View the virtual-host list.

   # **show virtual-host list**

   ```
   Currently configured virtual-hosts :
   Host              certificate    key             cipher
   ============================================
   10.157.34.129     pubs_doc     pubs_doc_key    N/A
   ------------------------------------------------------------------------------
   wildcard          N/A          N/A             N/A
   ------------------------------------------------------------------------------
   ```

   Use the **no virtual-host [wildcard|*virtual-domain-name*] certificate bind <*cert-name*> key bind <*key-name*>** CLI command to unconfigure the virtual-host certificate and key binding.

4. Exit enable CLI mode.

   # **exit**

Related Documentation

- HTTPS Support and SSL Client Delivery Overview on page 147
- Installing the Media Flow Controller SSL License on page 148
- Configuring the SSL Key File on page 151
- Configuring an SSL Certificate File on page 156
- Configuring the Virtual-Host Cipher String on page 161
- Mapping HTTPS Namespaces and Virtual Hosts on page 165

Configuring the Virtual-Host Cipher String

A cipher string is the data encryption specification. The support cipher strings are displayed in the virtual-host cipher string CLI command output.

During the SSL handshake, the SSL client announces the suite of ciphers that it supports in the configured order of preference. The SSL server then selects from the cipher list a cipher that matches its own list of configured ciphers. If the ciphers announced by the client do not match those configured on the SSL server, the SSL handshake fails.

To configure the virtual-host SSL cipher string:

1. Enter the CLI configure mode.

> **enable**

 # configure terminal

2. Configure one or more cipher strings for a virtual host.

   Enter multiple cipher strings separated by a colon (:). To negate a cipher, use an exclamation mark (!).

   (config) # **virtual-host <*name*> cipher <*cipher-string*>**

   For example, enter **virtual-host 10.157.34.129 cipher RC4-SHA:RC4-MD5**.

   The command **virtual-host <*name*> cipher** ? help text displays the following SSL cipher string list:

   NOTE: The list of cipher strings may or may not change for a Media Flow Controller release. Therefore, the example output shown here may not be complete.

```
DHE-RSA-AES256-SHA      Specify the cipher as DHE-RSA-AES256-SHA
DHE-DSS-AES256-SHA      Specify the cipher as DHE-DSS-AES256-SHA
AES256-SHA              Specify the cipher as AES256-SHA
EDH-RSA-DES-CBC3-SHA    Specify the cipher as EDH-RSA-DES-CBC3-SHA
EDH-DSS-DES-CBC3-SHA    Specify the cipher as EDH-DSS-DES-CBC3-SHA
DES-CBC3-SHA            Specify the cipher as DES-CBC3-SHA
DES-CBC3-MD5            Specify the cipher as DES-CBC3-MD5
DHE-RSA-AES128-SHA      Specify the cipher as DHE-RSA-AES128-SHA
DHE-DSS-AES128-SHA      Specify the cipher as DHE-DSS-AES128-SHA
AES128-SHA              Specify the cipher as AES128-SHA
RC2-CBC-MD5             Specify the cipher as RC2-CBC-MD5
RC4-SHA                 Specify the cipher as RC4-SHA
RC4-MD5                 Specify the cipher as RC4-MD5
EDH-RSA-DES-CBC-SHA     Specify the cipher as EDH-RSA-DES-CBC-SHA
EDH-DSS-DES-CBC-SHA     Specify the cipher as EDH-DSS-DES-CBC-SHA
DES-CBC-SHA             Specify the cipher as DES-CBC-SHA
DES-CBC-MD5             Specify the cipher as DES-CBC-MD5
EXP-EDH-RSA-DES-CBC-SHA Specify the cipher as EXP-EDH-RSA-DES-CBC-SHA
EXP-EDH-DSS-DES-CBC-SHA Specify the cipher as EXP-EDH-DSS-DES-CBC-SHA
EXP-DES-CBC-SHA         Specify the cipher as EXP-DES-CBC-SHA
EXP-RC2-CBC-MD5         Specify the cipher as EXP-RC2-CBC-MD5
EXP-RC4-MD5             Specify the cipher as EXP-RC4-MD5
```

   Use the **no virtual-host <name> cipher <cipher-string>** CLI command to remove the virtual-host cipher string configuration.

Related Documentation

- HTTPS Support and SSL Client Delivery Overview on page 147
- Installing the Media Flow Controller SSL License on page 148
- Configuring the SSL Key File on page 151
- Configuring an SSL Certificate File on page 156
- Binding an SSL Certificate and Key to a Virtual Host on page 160

- Mapping HTTPS Namespaces and Virtual Hosts on page 165

Enabling HTTP and HTTPS Client Delivery per Namespace

You can map an HTTPS namespace to multiple virtual hosts. For example :

```
virtual-host abc.com certificate bind cert1.pem key bind key1.pem
virtual-host xyz.com certificate bind cert2.pem key bind key2.pem
namespace ns_https
namespace ns_https domain abc|xyz.com
namespace ns_https match uri /
namespace ns_https origin-server http 172.19.159.20 80
namespace ns_https status active
```

Similarly, you can map a virtual host to multiple namespaces. For example:

```
virtual-host test.com certificate bind cert.pem key bind key.pem

namespace ns_https_1
namespace ns_https_1 match uri /pattern1
namespace ns_https_1 domain test.com

namespace ns_https_2
namespace ns_https_2 match uri /pattern2
namespace ns_https_2 domain test.com
```


- Enabling Only HTTPS Client Delivery on page 163
- Enabling Only HTTP Client Delivery (Default) on page 164
- Enabling HTTP and HTTPS Delivery on page 164

Enabling Only HTTPS Client Delivery

To enable HTTPS client delivery:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Reset the delivery options, client-side request parameters, and disable non-secure client delivery for a namespace.

   (config) # **no namespace <*name*> delivery protocol http client-request plain-text**

3. Enable HTTPS for the client.

   (config) #**namespace <*name*> delivery protocol http client-request secure**

4. View the namespace configuration:

   (config) # **show namespace <*name*>**

   ```
   . . .
    Client-Request Configuration:
      Secure Delivery(HTTPS over SSL):Enable
      Plain-text Delivery:Disable
      Allow objects with a query-string to be cached: no
   ```

```
Cache control max age : 0
Cache control max age action : serve-from-origin
Allow objects with a cookie header to be cached: yes
Revalidate object always: No
Tunnel All Requests : Disabled
Tunnel-Override Configuration:
Action on Connect method :  TUNNEL
```

Enabling Only HTTP Client Delivery (Default)

To enable HTTP client delivery:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Enable HTTP for the client.

   (config) # **namespace <name> delivery protocol http client-request plain-text**

3. Disable HTTPS for the client.

   (config) # **no namespace <name> delivery protocol http client-request secure**

4. View the namespace configuration:

   (config) # **show namespace *name***

```
...
 Client-Request Configuration:
  Secure Delivery(HTTPS over SSL):Disable
  Plain-text Delivery:Enable
  Allow objects with a query-string to be cached: no
  Cache control max age : 0
  Cache control max age action : serve-from-origin
  Allow objects with a cookie header to be cached: yes
  Revalidate object always: No
  Tunnel All Requests : Disabled
  Tunnel-Override Configuration:
  Action on Connect method :  TUNNEL
```

Enabling HTTP and HTTPS Delivery

To enable both HTTP and HTTPS client delivery:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Enable HTTP for the client.

   (config) # **namespace <*name*> delivery protocol http client-request plain-text**

3. Enable HTTPS for the client.

   (config) # **namespace <*name*> delivery protocol http client-request secure**

4. View the namespace configuration:

   (config) # **show namespace** *name*

   ```
   ...
    Client-Request Configuration:
     Secure Delivery(HTTPS over SSL):Enable
     Plain-text Delivery:Enable
     Allow objects with a query-string to be cached: no
     Cache control max age : 0
     Cache control max age action : serve-from-origin
     Allow objects with a cookie header to be cached: yes
     Revalidate object always: No
     Tunnel All Requests : Disabled
     Tunnel-Override Configuration:
     Action on Connect method :  TUNNEL
   ```

Related Documentation

- HTTPS Support and SSL Client Delivery Overview on page 147
- Configuring the HTTPS Delivery Protocol Interface on page 149
- Configuring HTTPS Protocol Listen Port Numbers on page 150
- Mapping HTTPS Namespaces and Virtual Hosts on page 165

Mapping HTTPS Namespaces and Virtual Hosts

You can map an HTTPS namespace to multiple virtual hosts. You can also map a virtual host to multiple namespaces.

1. Mapping an HTTPS Namespace to Multiple virtual hosts on page 165
2. Mapping a Virtual Host to Multiple HTTPS Namespaces on page 166

Mapping an HTTPS Namespace to Multiple virtual hosts

To map an HTTPS namespace to multiple virtual hosts:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Configure the following.

   For example:

   virtual-host *abc.com* **certificate bind** *cert1.pem* **key bind** *key1.pem*

   virtual-host *xyz.com* **certificate bind** *cert2.pem* **key bind** *key2.pem*

   namespace *ns_https*

   namespace *ns_https* **domain** *abc* | *xyz.com*

   namespace *ns_https* **match uri** */*

   namespace *ns_https* **origin-server http** *172.19.159.20 80*

namespace *ns_https* status active

Mapping a Virtual Host to Multiple HTTPS Namespaces

To map a virtual host to multiple HTTPS namespaces:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Configure the following.

   For example:

   virtual-host *test.com* certificate bind *cert.pem* key bind *key.pem*

   namespace *ns_https_1*

   namespace *ns_https_1* match uri /pattern1

   namespace *ns_https_1* domain *test.com*

   namespace *ns_https_2*

   namespace *ns_https_2* match uri /pattern2

   namespace *ns_https_2* domain *test.com*

Related Documentation

- Configuring the SSL Key File on page 151
- Configuring the SSL CSR File on page 154
- Configuring an SSL Certificate File on page 156
- Binding an SSL Certificate and Key to a Virtual Host on page 160
- Configuring the Virtual-Host Cipher String on page 161
- Enabling HTTP and HTTPS Client Delivery per Namespace on page 163

Configuring HTTPS Fetch from Origin Servers

If you want to crawl and download content from origin servers that reside within non-trusted domains, the parent Media Flow Controller can be configured to authenticate the origin servers directly before crawling or downloading of content can begin. Media Flow Controller supports NULL-MD5 and NULL-SHA cipher suites for origin server authentication. When these cipher suites are used, Media Flow Controller encrypts the authentication messages and the content is not encrypted.

A list of trusted Certification Authority (CA) certificates is stored in the parent cache. CAs must be preloaded into the Media Flow Controller to authenticate with the origin server. Using the CLI, you can upload a list of root certificates for the CAs before downloading any data from origin servers. The handshake protocol uses this certificate to provide a digital form of identification, containing identification information, a valid period, a public key, and the digital signature of the issuer. The certificate binds certificate holder's identity to the public key.

To authenticate origin servers:

1. Create a namespace.

   namespace *name* origin-server http *origin-ip-address/domain-name port-number*

   The default origin server HTTP port number is 80, but for HTTPS, use port number 443.

2. Enable the SSL origin server to download within a namespace. A secure connection must be requested from the origin server and enabled for the namespace for which you want to use SSL origin server download.

   namespace *name* delivery protocol http origin-request secure

3. Configure the SSL no encryption cipher text.

   ssl cipher eNULL

   eNULL is equivalent to NULL-SHA:NULL-MD5. The CLI command option for SSL_RSA_WITH_NULL_MDS is NULL-MD5 and for SSL_RSA_WITH_NULL_SHA the command option is NULL-SHA Media Flow Controller supports ciphers for NULL encryption using cipher suites. For more information, see the *Media Flow Controller CLI Command Reference*.

4. Import the trusted Certificate Authority (CA) certificates.

   ssl ca-certificate *ca-certificate-name* import *scp://username:password@hostname/path*

5. Enable SSL server authentication. Media Flow Controller provides global-level settings to enable and disable server authentication.

   ssl server-authentication enable

Related Documentation

- HTTPS Support and SSL Client Delivery Overview on page 147
- Installing the Media Flow Controller SSL License on page 148

Verifying HTTPS Client Delivery

- Viewing HTTPS Performance Counters on page 167
- Viewing HTTPS Client Delivery Access Logs on page 168
- Viewing Error Logs for SSL Handshake Failures on page 169

Viewing HTTPS Performance Counters

Media Flow Controller provides counters for HTTPS performance.

To view HTTPS performance:

1. View HTTPS performance counters.

   > **show counters https**

   The following counters appear:

```
Total number of Client HTTPS Connections       : 18051
Total number of Client HTTPS Transactions      : 0
Total number of Client Open HTTPS Connections  : 0
Total number of Bytes sent to Client           : 0
Total number of Bytes received from Client     : 0
Total number of Client handshake success Count  : 0
Total number of Origin Current Open Sockets    : 0
Total number of Origin  Sockets                : 18051
Total number of bytes received from Origin     : 284794398
Total number of bytes sent to Origin           : 4317525
Total number of Origin handshake failures      : 4
Total number of Origin handshake Success Count  : 17989
```

Table 19 on page 168 describes the Media Flow Controller HTTPS client delivery performance counters.

Table 19: HTTPS Client Delivery Counters

Counter Name	Description
Total number of HTTPS Connections	The amount of HTTPS connections made to Media Flow Controller.
Total number of HTTPS Transactions	The amount of HTTPS transactions Media Flow Controller received.
Total number of Open HTTPS Connections	The amount of open HTTPS connections to Media Flow Controller.
Total number of Bytes sent from Client	The amount of HTTPS bytes Media Flow Controller sent.
Total number of Bytes received from Client	The amount of HTTPS bytes Media Flow Controller received.
Total number of Client handshake success Count	The amount of handshakes completed without errors.
Total number of Origin Current Open Sockets	The amount of open HTTPS connections from Media Flow Controller to the origin.
Total number of Origin Sockets	The amount of HTTPS connections made from Media Flow Controller to the origin.
Total number of bytes received from Origin	The amount of HTTPS bytes Media Flow Controller received from origin.
Total number of bytes sent to Origin	The amount of HTTPS bytes Media Flow Controller sent to origin.
Total number of Origin handshake failures	The amount of origin side SSL handshake failures.
Total number of Origin handshake Success Count	The amount of succeeded origin side SSL handshakes

Viewing HTTPS Client Delivery Access Logs

You can log the HTTPS delivery port number using %Z access log format option to differentiate the HTTPS requests in access logs.

To view HTTPS access logs:

1. **show accesslog <*filename*>**

   ```
   100.1.1.68 - - [05/Feb/2012:11:59:48 +0000] GET
   /ssl/1MB1266OKFQKS27857187307737eiLal872855897840804
   967910963NtMNAW07907205502716pPO0153QGAQCD2822713
   5435730XRD2320Sxj/C1P9w4h6Q8C4e3Q2H4X9Q11
   HTTP/1.1 200 10485760
   ```

Viewing Error Logs for SSL Handshake Failures

If the SSL handshake fails, Media Flow Controller logs an error description in **error.log**.

To view error logs:

1. View the error logs:

   # show errorlog (last | continuous) <*number of lines*>

   ```
   [Wed Nov 23 01:56:24.001 2011][MOD_SSL.MSG] ssl_accept:642: ssl_accept:
   r_code=-1 , ssl_get_error 1
   [Wed Nov 23 01:56:24.001 2011][MOD_SSL.WARNING] ssl_accept:659:
   error:1408A0C1:SSL routines:SSL3_GET_CLIENT_HELLO:no shared
   cipher:s3_srvr.c:1221:

   Ø  [Wed Nov 23 02:02:47.902 2011][MOD_SSL.WARNING] ssl_accept:659:
   error:14094418:SSL routines:SSL3_READ_BYTES:tlsv1 alert unknown
   ca:s3_pkt.c:1193:SSL alert number 48
   ```

Related Documentation

- HTTPS Support and SSL Client Delivery Overview on page 147
- Configuring the HTTPS Delivery Protocol Interface on page 149
- Configuring HTTPS Protocol Listen Port Numbers on page 150

CHAPTER 6

Content Ingest Manager

Getting Started and Configuration

Content Ingest Manager Overview

The Content Ingest Manager allows you to keep your content fresh by periodically searching content provider repositories or parent caches and downloading fresh content to your parent and edge Media Flow Controller caches using a scheduled HTTP crawler program. The following are the benefits of Content Ingest Manager:

- Automatic download of content from origin file servers at scheduled intervals. It is not necessary for you to initiate the propagation of content across the Content Delivery Network (CDN).
- You are always served fresh content. Cached content is kept fresh by periodically refreshing it from origin file servers.
- Almost all the requests for content are served from the edge cache to improve the user experience

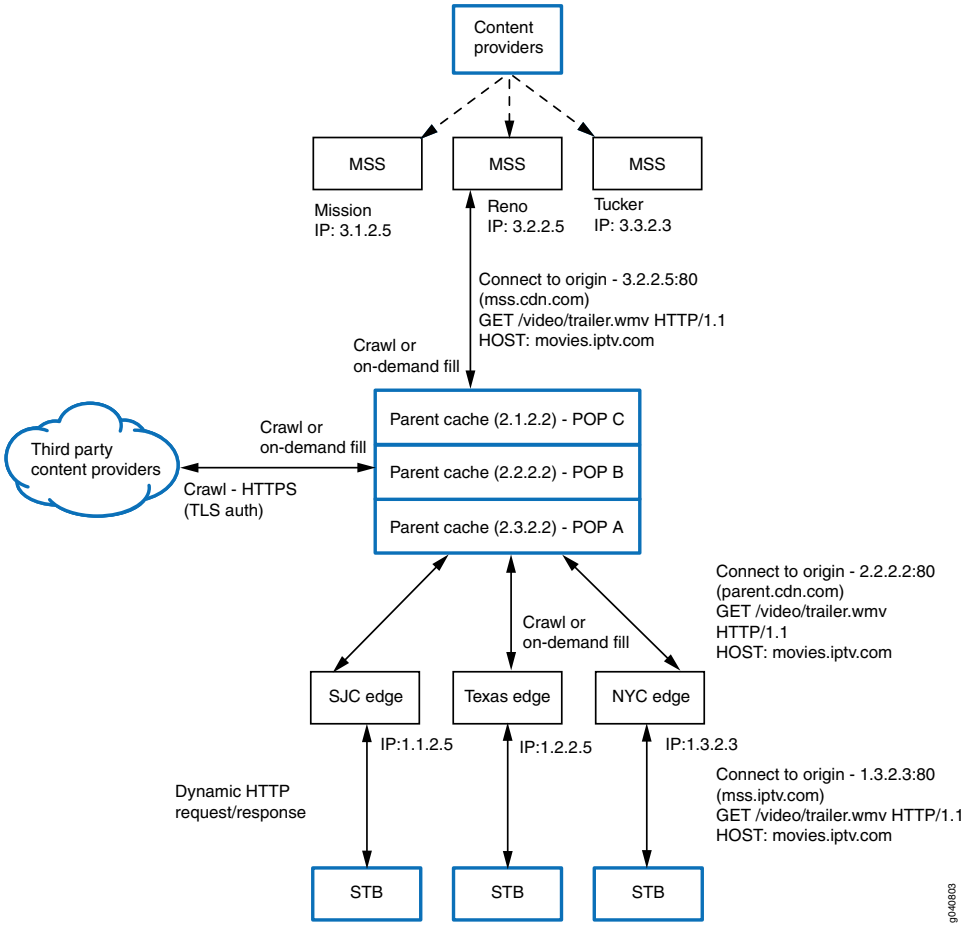
Media Flow Controller supports crawling of both directory listing and web pages hosted by a web server. For directory list crawl, Content Ingest Manager uses the depth configuration to traverse the specified number of directories. For web page crawl, Content Ingest Manager uses the depth configuration to traverse the URL links in a web page.

Parent caches crawl Media Staging Servers (MSSs), which are repositories where content providers store permanent content, or other origin servers and download content at

scheduled intervals. Edge caches, in-turn, crawl parent caches and download content at scheduled intervals.

The crawler is recursive and starts discovering URLs from the origin base URL, downloading the discovered URLs from the origin. Media Flow Controller discovers content on origin servers and pulls down content that matches predefined filtering rules. Eventually, all the parent caches running Media Flow Controller software will have completed downloading content from the MSS origin based on the configured content ingest policies. As the edge caches crawl the parent caches, content from the origin servers is then transferred to the edge caches running standard Media Flow Controller software (see Figure 7 on page 172). From the edge caches, it is served HTTP to clients running on set-top boxes (STBs).

Figure 7: Content Ingest Manager



The Content Ingest Manager can be used in CDN or IPTV deployments. It is used for fetching content from the origin file server and to propagate content across the entire CDN. The Content Ingest Manager allows pinning of the crawled content in the Media Flow Controller cache and provides additional functions for Windows Media Services such as Advanced Stream Redirector (ASX) file generation. For every Windows Media Video (WMV) file that is fetched from origin, Media Flow Controller can be configured to generate an ASX file. Advanced Systems Format (formerly, Advanced Streaming Format;

Active Streaming Format) ASF files, which contain the actual video and audio media, are used to point to and start the ASF file streaming; they are equivalent to a playlist or redirector and are not published by content providers.

With certain applications, a set-top box running Windows Media Player (WMP) can request an Advance Stream Redirector (ASX) file that contains the link to the video to be played. An ASX file is equivalent of a playlist or redirector file and is not published by the content provider. Media Flow Controllers can be configured to generate ASX files.

The parent cache selects the origin MSS, by resolving the Fully Qualified Domain Name (FQDN) to one or more IP addresses of the origin servers. Media Flow Controller performs origin escalation when any of the origin servers are not available for service.

Media Flow Controller can be configured to authenticate the origin server before downloading content. When origin server authentication is enabled, SSL 3.0 or TLS 1.0 authentication occurs over a secure channel and content is transferred without encryption (in plain text). Media Flow Controller does not support client authentication with the origin server.

If the origin server responds with a **cache-control=no-cache** for a crawl request from Media Flow Controller, then it caches the object in its cache and marks it for **revalidation on every client request**. While caching, Media Flow Controller removes the **no-cache** directive and replaces it with **max-age=<value** configured for the **cache-age-default>** directive. This directive is served to clients (end clients or another edge Media Flow Controller) throughout the life time of the object. Therefore, if a request hits Media Flow Controller towards the end of the object life, it serves the data with **max-age=<cache-age-default>**, and this action may create a scenario where another cache between the client and the Media Flow Controller to hold the object for another **cache-age-default** seconds. This behavior is visibly seen by the client in case of a one-time crawl scheme only. In case of a synchronized edge parent crawl setup with periodic refresh scheme, this behavior is not visible to the client.

Because set-top boxes running Windows Media Player (WMP) request ASX files, which contain a link to the video content to be played, you might want to configure ASX file generation under a crawler instance .

When ASX file generation is enabled under the crawler definition, Media Flow Controller generates an ASX file for every WMV file that is crawled from the origin server. ASX file generation occurs before the corresponding file is preloaded from the origin. The parent Media Flow Controller crawler always generates ASX files for WMV files. The edge Media Flow Controllers can be configured to generate ASX files at the edge for crawled content or populated by cache-miss fetches from parent Media Flow Controller. Edge Media Flow Controllers can also be configured so that they do not download WMV objects and only generate ASX files. Service providers can populate edge caches for select namespaces only when a user makes requests for content.

For delivery of ASX (and WMV) files, the HREF in the generated ASX file uses the same domain name as the URL configured in the crawler profile. If you do not want to use the same domain name, it can be overridden by configuring the domain name under the crawler profile. The domain name configuration in the crawler profile is used to create the WMV link in the ASX file. ASX files generated by the crawler do not expire. ASX files

are removed from the cache when the corresponding WMV file is removed by Media Flow Controller during the next crawl operation or when an explicit delete for ASX file is issued using the command line interface.

ASX files generated by the crawler are automatically pinned to the cache file system. If pinning is enabled under a namespace, those ASX files populated by a cache-miss fetch at the edge Media Flow Controller are also pinned to the cache file system.

Related Documentation

- Setting Up the Content Ingest Manager on page 174
- Configuring HTTPS Fetch from Origin Servers on page 166.
- Viewing the Status of the HTTP Crawler on page 182
- Viewing HTTP Crawler Instance on page 182
- Viewing the Pinning Status in Cache of Namespace on page 183

Setting Up the Content Ingest Manager

The Content Ingest Manager allows you to keep your content fresh by periodically searching content provider repositories and downloading fresh content to your parent and edge Media Flow Controller caches.

You can configure the Content Ingest Manager crawler to search Media Staging Server repositories or other origin servers to download fresh content to the Media Flow Controller parent cache at scheduled intervals. You can configure Media Flow Controller edge caches to crawl the parent caches at scheduled intervals and, in turn, download fresh content. The crawling process discovers content on the origin servers and downloads content that matches established filtering rules.

When you configure a crawler, the first step is to determine the URL of the content repository (that is, the path to the content) that you want to crawl and then associate this URL with the crawler. The corresponding CLI command is as follows: **crawler *<crawler-name>* base-url *<URL>*** (for example, **crawler test base-url http://www.origin.com/**). In this example, the "test" crawler crawls and caches content from the origin.com server.

The next step is to determine the type of crawler that you want to create. There are two types of crawlers: directory-based crawler and webpage-based crawler.

- Create a directory-based crawler when you want the Media Flow Controller to crawl directories.

  To create a directory-based crawler, enter the URL of the directory in the base URL configuration. For example, when you enter http://www.origin.com/video/ in the base URL configuration, the crawler crawls the "video" directory and caches its contents and subdirectories based on the link-depth configuration of the crawler.

- Create a webpage-based crawler when you want the Media Flow Controller to crawl an HTML file.

  To create a webpage-based crawler, you provide a path to an HTML file in the base URL configuration. For example, when your enter http://www.origin.com/page/foo.html

in the base URL configuration, the crawler caches this HTML file and any other HREF links within this file based on the link-depth configuration of the crawler.

The next step is to decide whether you want to cache the subdirectories within a directory (in a directory-based crawler) or the links within an HTML file (in a webpage-based crawler).

To cache subdirectories within a directory:

Consider the website, http://www.origin.com/video/, which contains several Flash video files (*.flv files) that belong to the current year (such as **worldtoday.flv** and **latestgadgets.flv**) and a few directories or folders labeled by year containing videos pertaining to that year, such as 2011 and 2010 directories containing videos pertaining to 2011 and 2010, respectively. The link-depth configuration plays a major role in determining whether the objects from the subdirectories (in this example, whether videos from the 2011 and 2010 directories) are to be cached or not. If you set the link-depth to 0 (zero), the Media Flow Controller caches only the objects directly under the base URL and not under the subdirectories. Any subdirectory within the base URL is considered at link-depth 1. By default, the link-depth is set to 10; however, the link-depth can be configured to any value between 0 and 10.

So in this example, if you configure the base URL as http://www.origin.com/video/ and the link-depth as 0, the Media Flow Controller caches only **worldtoday.flv** and **latestgadgets.flv** objects and not the objects within the 2011 and 2010 directories. There are many ways by which you can cache objects within these directories. Two examples are:

- Setting the base URL to http://www.origin.com/video/ and the link-depth to 1. Here, all the objects within the "video" folder are cached, including objects in the 2011 and 2010 directories respectively.
- Setting the base URL to http://www.origin.com/video/2011 and the link-depth to 0. Here, only the objects within the 2011 directory are cached and not the subdirectories within it.

In a nutshell, if the base URL is http://www.origin.com/video/:

- Files and subdirectories under "**/video/**" are considered at link-depth 0
- Files and subdirectories under **"/video/2011**" are considered at link-depth 1
- Files and subdirectories under **"/video/2011/Jan**" are considered at link-depth 2

To cache subdirectories within the 2011 and 2010 directories, using the base URL http://www.origin.com/video/, set the link-depth as 2. However, if you want to cache videos taken in January 2011 only, provide the base URL http://www.origin.com/video/2011/Jan and set the link-depth to 0.

To summarize, using a base URL of **/dir1/dir2/dir3/dir4/dir5/dir6/** and a link-depth of 0, the Media Flow Controller caches all the objects within **dir6** only, without caching the subdirectories within dir6. Using a base URL of **/dir1/** and a link-depth of 5, the Media

Flow Controller caches all the objects within **dir1**, **dir2**, **dir3**, **dir4**, **dir5**, and **dir6**, without caching the subdirectories within **dir6**.

To cache links within an HTML file:

Consider how the Media Flow Controller crawler handles a base URL of http://www.origin.com/foo.html with a link-depth of 0. Because the links within this HTML file (foo.html) are at a link depth of 0, the crawler caches the HTML file along with the objects referenced by a URL under this HTML file. But if a URL under this HTML file points to a directory, the objects under that directory are not cached. So to cache objects within the sublinks of this HTML file, set the link-depth as 1. There may be other links within each of these sublinks. These links are considered at a link-depth of 2.

To summarize, if an HTML file is available at level N, then all the links under that HTML file are considered to be at a link-depth of N+1. If N and N+1 is less than or equal to the link-depth configured in the crawler, then all the objects within the HTML file is cached.

You can configure the Media Flow Controller to authenticate the origin servers directly before crawling or downloading of content can begin. Typically, origin servers that reside outside the CDN's trusted domain are authenticated. For information on configuring Media Flow Controller to authenticate origin servers, see "Configuring HTTPS Fetch from Origin Servers" on page 166.

To configure your Content Ingest Manager HTTP crawler on either a Media Flow Controller parent or edge cache:

1. Create a crawler instance by specifying the origin server protocol, URL, and port number.

   - Origin URLs must be configured such that they do not overlap. For example, do not configure http://www.origin.com/example/video/ and http://www.origin.com/example in the same Media Flow Controller. Configuring just http://www.origin.com/example is sufficient.
   - Specify origin servers by using hostnames or IP addresses.

   crawler ***crawler-name*** **base-url** ***url***

2. Set up the crawl schedule, where the refresh Interval is 0 or equal to or greater than five minutes entered in one-minute increments.

   - Crawler begins crawling at the specified start time.
   - After the stop time, new web crawling sessions are not started. If a crawling operation is in progress after the stop time is reached, the crawling operation is gracefully completed.
   - If a stop time is not specified, the crawler automatically continues to crawl at the specified refresh intervals.

   crawler ***crawler-name*** **schedule start** ***yyyy/mm/dd hh:mm:ss*** **stop** ***yyyy/mm/dd hh:mm:ss*** **refresh-interval** ***interval-minutes***

NOTE: All Media Flow Controllers must be set to the Coordinated Universal Time (UTC) standard for the crawler to work properly. If the Media Flow Controllers are not set to UTC, crawler operation and results will be unpredictable.

Additionally, both *yyyy/mm/dd* and *hh:mm:ss* arguments are required for start and stop times. If the refresh interval is set to 0, the crawler does not start again after the first crawl period.

NOTE: If a crawler is originally scheduled with a start time and stop time, and if you later want to schedule it as a nonstop crawler, the stop time should be set to 1970/01/01 00:00:00. This removes the crawler's stop time.

For example, if a crawler schedule is configured using the command, **crawler test schedule start 2012/02/09 03:00:50 stop 2012/02/09 03:30:00 refresh-interval 10**, you can remove the stop time such that the crawler test schedule start time is 2012/02/09 03:00:50 and stop time is 1970/01/01 00:00:00 with a refresh-interval of 10.

To remove the stop time, use the command: **crawler test schedule start 2012/02/09 03:00:50 and stop 1970/01/01 00:00:00 refresh-interval 10.**

After configuring the crawler schedule using these two example commands, the crawler behaves as if you only configured the start time: **crawler test schedule start 2012/02/09 03:00:50 refresh-interval 10.**

If you configure the crawler schedule with a past start time, Media Flow Activate provisioning occurs successfully; . However, if you configure the crawler schedule with a past start time using the Media Flow Controller CLI, a **configured time occurs in the past** warning appears.

3. Configure the crawler directory crawl depth level from 1 to N, where N=10.

   - The default crawl directory depth limit is 10 levels deep. A selected crawl directory depth of 0 means that only the object at the base URL, which might be a directory, is downloaded.

     For example, assuming that http://www.origin.com/example is the base-url and 3 is the depth level, the following are the various directories that are crawled:

     http://www.origin.com/example - depth 0
     http://www.origin.com/example/e1/ - depth 1
     http://www.origin.com/example/e1/e2/ - depth 2
     http://www.origin.com/example/e1/e2/e3 - depth 3

   crawler *crawler-name* link-depth [0-N]

4. Define the file extensions to be accepted and downloaded, where a multiple of up to 10 instances is supported.

When defining file extensions, specify:

- Only one file extension at a time.
- Up to 10 file extensions.
- Accept all static content.
- Only dotted file extensions of the form ".xxx" (for example ".wmv").
- Optional. The **skip-preload** option is used if you do not want the crawler to perform preload of the objects found that match this file extension. When combined with the auto-generate command, you can get Advanced Stream Redirector (ASX) file generation without preload of the WMV file.

crawler *crawler-name* accept-file-extension *extension-string* [skip-preload]

5. Enable the crawler.

   crawler *crawler-name* status active

6. Disable the crawler.

   crawler *crawler-name* status inactive

7. Configure cache capacity, in megabytes, per namespace.

   NOTE: Cache capacity can only be configured for the lowest Media Flow Controller cache tier.

   - The total size limit can be configured.
   - The configured size limit is for a single tier, where only one tier can be limited at any one time per namespace.

   namespace *name* resource-limit cache-tier [SAS | SATA | SSD] cache-capacity *capacity*

You can configure ASX file generation under each crawler instance for both the Media Flow Controller parent and edge crawlers. Once generated, these files behave like other cached objects in Media Flow Controller. When a Windows Media Video (WMV) file is deleted during a crawl session and the WMV file no longer exists at the origin, the corresponding ASX metafile is also removed from the cache. However, when the WMV file is manually removed from the CLI, the administrator must also explicitly remove the corresponding ASX file.

To optionally configure ASX file generation.

1. Optionally, enable ASX file generation for the defined crawler session:

   crawler *crawler-name* action auto-generate *string* source-file *string*

   There is a command option to enable generation of ASX files even though the corresponding WMV files are not downloaded to the cache. If the **skip-preload** option is selected using the **accept-file-extension** command, then a preload is not performed for objects in the current crawl session.

2. Optional. Add the automatic cache-pin attribute to pin every object to this namespace.

   namespace *name* pinned-object auto-pin

   In addition, type the following command to enable pinning:

   namespace *name* pinned-object status enable

The following are examples of crawler and namespace configurations for both edge and parent Media Flow Controllers. The edge configuration namespace and crawler names are **crawl1** and **crawl1**, respectively, and the parent configuration namespace and crawler names are **origin_crawl1** and **origin_crawl1**, respectively.

The DNS configuration is as follows:

- *dal1-1.scej.com* resolves to the edge Media Flow Controller IP address 21.127.252.66.
- *parent1.scej.com* resolves to the parent Media Flow Controller IP address 21.127.250.98.
- *joust-inet-web1.scej.com* resolves to the apache webserver IP address 21.127.250.40.

NOTE: On the edge Media Flow Controller, the namespace points to its respective parent's namespace domain as its origin server **parent1.scej.com**.

Edge Media Flow Controller Configuration Example

Edge namespace **crawl1**:

```
namespace crawl1 accesslog default
  namespace crawl1 cluster-hash complete-url
  namespace crawl1 delivery protocol http cache-index domain-name include
 namespace crawl1 delivery protocol http client-request cache-control max-age 0 action
serve-from-origin
  namespace crawl1 delivery protocol http client-request cookie action cache
  namespace crawl1 delivery protocol http client-request geo-service lookup-timeout
5000
  namespace crawl1 delivery protocol http client-request plain-text
  namespace crawl1 delivery protocol http client-request query-string action no-cache
  namespace crawl1 delivery protocol http connection concurrent session retry-after 0
  namespace crawl1 delivery protocol http origin-fetch cache-age-default 600
  namespace crawl1 delivery protocol http origin-fetch cache-control header ""
  namespace crawl1 delivery protocol http origin-fetch cache-directive no-cache follow

  namespace crawl1 delivery protocol http origin-fetch cache-fill client-driven
  namespace crawl1 delivery protocol http origin-fetch cache-fill client-driven
aggressive-threshold 9
  namespace crawl1 delivery protocol http origin-fetch cache-ingest hotness-threshold
 3
  namespace crawl1 delivery protocol http origin-fetch content-store media
cache-age-threshold 60
  namespace crawl1 delivery protocol http origin-fetch content-store media cache-tier
highest object-size 0
  namespace crawl1 delivery protocol http origin-fetch content-store media object-size
 4096
```

```
  namespace crawl1 delivery protocol http origin-fetch content-store media
uri-depth-threshold 10
  namespace crawl1 delivery protocol http origin-fetch header set-cookie any cache deny

  namespace crawl1 delivery protocol http origin-fetch packing-policy fifo
  namespace crawl1 delivery protocol http origin-fetch redirect-302 pass-through
  namespace crawl1 delivery protocol http origin-request cache-revalidation permit
  namespace crawl1 delivery protocol http origin-request cache-revalidation permit
method head
  namespace crawl1 delivery protocol http origin-request host-header inherit incoming-req
 permit
  namespace crawl1 delivery protocol http origin-request x-forwarded-for enable
  namespace crawl1 delivery protocol rtsp cache-index domain-name include
  namespace crawl1 delivery protocol rtsp origin-fetch cache-age-default 0
  namespace crawl1 domain dal1-1.scej.com
  namespace crawl1 match uri /Video1
  namespace crawl1 media-cache disk cache-tier sas free-block-threshold 50
  namespace crawl1 media-cache disk cache-tier sas group-read enable
  namespace crawl1 media-cache disk cache-tier sas read-size 2048
  namespace crawl1 media-cache disk cache-tier sata free-block-threshold 50
  namespace crawl1 media-cache disk cache-tier sata group-read enable
  namespace crawl1 media-cache disk cache-tier sata read-size 2048
  namespace crawl1 media-cache disk cache-tier ssd free-block-threshold 50
  namespace crawl1 media-cache disk cache-tier ssd read-size 256
  namespace crawl1 origin-server http "parent1.scej.com"
  namespace crawl1 pinned-object auto-pin
  namespace crawl1 pinned-object status enable
  namespace crawl1 resource-limit cache-tier sas cache-capacity 0
  namespace crawl1 resource-limit cache-tier sata cache-capacity 0
  namespace crawl1 resource-limit cache-tier ssd cache-capacity 0
  namespace crawl1 status active
  namespace crawl1 virtual-player ""
```

Edge crawler **crawl1**:

```
crawler crawl1 base-url http://dal1-1.scej.com/Video1
  crawler crawl1 link-depth 3
  crawler crawl1 schedule start 2012/02/07 16:55:00 stop 2012/02/26 19:25:00
refresh-interval 10
  crawler crawl1 accept-file-extension .wmv
  crawler crawl1 status active
```

Parent Media Flow Controller Configuration Example

Parent namespace **origin_crawl**:

```
namespace origin_crawl1 accesslog default
  namespace origin_crawl1 cluster-hash complete-url
  namespace origin_crawl1 delivery protocol http cache-index domain-name include
  namespace origin_crawl1 delivery protocol http client-request cache-control max-age
 0 action serve-from-origin
  namespace origin_crawl1 delivery protocol http client-request cookie action cache
  namespace '/;origin_crawl1 delivery protocol http client-request geo-service
lookup-timeout 5000
  namespace origin_crawl1 delivery protocol http client-request plain-text
  namespace origin_crawl1 delivery protocol http client-request query-string action
no-cache
```



```
 namespace origin_crawl1 delivery protocol http connection concurrent session retry-after
0
 namespace origin_crawl1 delivery protocol http origin-fetch cache-age-default 600
 namespace origin_crawl1 delivery protocol http origin-fetch cache-control header ""
 namespace origin_crawl1 delivery protocol http origin-fetch cache-directive no-cache
follow
 namespace origin_crawl1 delivery protocol http origin-fetch cache-fill client-driven
 namespace origin_crawl1 delivery protocol http origin-fetch cache-fill client-driven
aggressive-threshold 9
 namespace origin_crawl1 delivery protocol http origin-fetch cache-ingest
hotness-threshold 3
 namespace origin_crawl1 delivery protocol http origin-fetch content-store media
cache-age-threshold 60
 namespace origin_crawl1 delivery protocol http origin-fetch content-store media
cache-tier highest object-size 0
 namespace origin_crawl1 delivery protocol http origin-fetch content-store media
object-size 4096
 namespace origin_crawl1 delivery protocol http origin-fetch content-store media
uri-depth-threshold 10
 namespace origin_crawl1 delivery protocol http origin-fetch header set-cookie any cache
deny
 namespace origin_crawl1 delivery protocol http origin-fetch packing-policy fifo
 namespace origin_crawl1 delivery protocol http origin-fetch redirect-302 pass-through

 namespace origin_crawl1 delivery protocol http origin-request cache-revalidation permit

 namespace origin_crawl1 delivery protocol http origin-request cache-revalidation permit
method head
 namespace origin_crawl1 delivery protocol http origin-request host-header inherit
incoming-req deny
 namespace origin_crawl1 delivery protocol http origin-request x-forwarded-for enable

 namespace origin_crawl1 delivery protocol rtsp cache-index domain-name include
 namespace origin_crawl1 delivery protocol rtsp origin-fetch cache-age-default 0
 namespace origin_crawl1 domain dal1-1.scej.com
 namespace origin_crawl1 match uri /Video1
 namespace origin_crawl1 media-cache disk cache-tier sas free-block-threshold 50
 namespace origin_crawl1 media-cache disk cache-tier sas group-read enable
 namespace origin_crawl1 media-cache disk cache-tier sas read-size 2048
 namespace origin_crawl1 media-cache disk cache-tier sata free-block-threshold 50
 namespace origin_crawl1 media-cache disk cache-tier sata group-read enable
 namespace origin_crawl1 media-cache disk cache-tier sata read-size 2048
 namespace origin_crawl1 media-cache disk cache-tier ssd free-block-threshold 50
 namespace origin_crawl1 media-cache disk cache-tier ssd read-size 256
 namespace origin_crawl1 origin-server http "joust-inet-web1.scej.com"
 namespace origin_crawl1 pinned-object auto-pin
 namespace origin_crawl1 pinned-object status enable
 namespace origin_crawl1 resource-limit cache-tier sas cache-capacity 0
 namespace origin_crawl1 resource-limit cache-tier sata cache-capacity 0
 namespace origin_crawl1 resource-limit cache-tier ssd cache-capacity 0
 namespace origin_crawl1 status active
 namespace origin_crawl1 virtual-player ""
```

Parent crawler **origin_crawl**:

```
crawler origin_crawl1 origin-server base-url http://dal1-1.scej.com/Video1
  crawler origin_crawl1 link-depth 3
  crawler origin_crawl1 schedule start 2012/02/07 12:30:00 stop 2012/02/26 11:40:00
refresh-interval 10
  crawler origin_crawl1 accept-file-extension .wmv
  crawler origin_crawl1 status active
```

Related Documentation

- Content Ingest Manager Overview on page 171
- Configuring HTTPS Fetch from Origin Servers on page 166.
- Viewing the Status of the HTTP Crawler on page 182
- Viewing HTTP Crawler Instance on page 182
- Viewing the Pinning Status in Cache of Namespace on page 183
- Viewing SSL Settings on page 184

Viewing the Status of the HTTP Crawler

Purpose View the status of crawler instances.

Action Run **show crawler list** to display the instance name and state.

```
MFC # show crawler list

Configured Crawler instances :
                Name : Status

              crawl1 : Inactive (Next crawl time: 2012/04/01 01:03:00 )
              crawl2 : Active (CRAWL_IN_PROGRESS)
           dualcrawl : Inactive (CRAWL_STOPPED)
          iis_crawl1 : Active (CRAWL_COMPLETED)
           longcrawl : Inactive (Next crawl time: 2012/04/01 01:05:00 )
```

Meaning Displays the status of crawler instances.

Related Documentation

- Content Ingest Manager Overview on page 171
- Setting Up the Content Ingest Manager on page 174
- Viewing HTTP Crawler Instance on page 182
- Viewing the Pinning Status in Cache of Namespace on page 183

Viewing HTTP Crawler Instance

Purpose View base URL, directory crawl depth, refresh interval, capacity limit, crawler start and stop timestamps, number of objects crawled, and crawling status of a crawler instance.

Action Run **show crawler *crawler-name*** to view base URL, directory crawl depth, refresh interval, capacity limit, and status of a crawler instance:

```
MFC # show crawler origin_crawl1
Crawler instance: origin_crawl1
Base URL: http://dal1-1.scej.com/Video1
```

```
Depth: 3
Refresh interval: 0
Status: inactive
  Preloaded file extensions:
    .wmv
  Non-preloaded file extensions:
Auto-generate: None
Scheduled start time: 2012/03/16 18:35:00
Scheduled stop time: 1970/01/01 00:00:00
Next crawl time: 2012/04/23 18:50:01
  Last Crawl Details:
start timestamp: None
end timestamp: None
Number of objects added: 0
Number of objects deleted: 0
Number of adds failed: 0
Number of deletes failed: 0
```

Meaning The output displays the configuration attributes of the specified crawler instance, including the base URL, directory crawl depth, refresh interval, crawl capacity limit, crawler start and stop timestamps, and crawling status.

Related Documentation

- Content Ingest Manager Overview on page 171
- Setting Up the Content Ingest Manager on page 174
- Viewing the Status of the HTTP Crawler on page 182
- Viewing the Pinning Status in Cache of Namespace on page 183

Viewing the Pinning Status in Cache of Namespace

Purpose View the object pinning status of objects in cache for specific namespace.

Action Run **show namespace list object all** to view the object pinning status.

```
MFC (config) # show namespace list object all

Objects in cache for namespace : Barbie

 *  Loc  Pinned  Size (KB)       Expiry                          URL
    dc_1    Y          1024  Wed Dec 22 18:19:29 2011
10.2.1.101:80/satis/1mb/1mb_8.wmv
    dc_3    Y          1024  Wed Dec 22 18:19:29 2011
10.2.1.101:80/satis/1mb/1mb_9.wmv
     :     :           :              :                                :
     :
Total object count: 924
Total object size: 109 GB
MFC (config) #
```

Meaning Displays the object pinning status of objects in cache for a specific namespace.

Related Documentation

- Content Ingest Manager Overview on page 171
- Setting Up the Content Ingest Manager on page 174

- Viewing the Status of the HTTP Crawler on page 182
- Viewing HTTP Crawler Instance on page 182

Viewing SSL Settings

Purpose To view the SSL Certificate Authority (CA) certificate, SSL certificate, and a list of configured ciphers.

Action Run the **show ssl ca-certificate *cert-name*** CLI command to view the CA certificate.

Run the **show ssl certificate *cert-name*** CLI command to view the SSL certificate.

Run the **show ssl cipher** CLI command to view the list of configured ciphers.

Meaning The individual show SSL command outputs display the configuration attributes of the specified SSL certificate or configured ciphers. For example:

Example: **show ssl ca-certificate *cert-name***

```
MFC (config) # show ssl ca-certificate ca.pem
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      d3:b7:07:35:8b:e8:5a:15
    Signature Algorithm: sha1WithRSAEncryption
    Issuer: C=IN, ST=Tamilnadu, L=Chennai, O=Ankeena, OU=Juniper,
CN=tester.juniper.net/emailAddress=tester@juniper.net
    Validity
      Not Before: Aug  8 17:36:01 2011 GMT
      Not After : Aug  5 17:36:01 2021 GMT
    Subject: C=IN, ST=Tamilnadu, L=Chennai, O=Ankeena, OU=Juniper,
CN=tester.juniper.net/emailAddress=tester@juniper.net
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (1024 bit)
        Modulus (1024 bit):
          00:c9:52:20:a1:e5:46:6b:d8:eb:73:bb:a0:67:ac:
          a8:92:90:34:f6:d6:7c:14:04:71:d1:ee:7b:60:e2:
          98:9f:9c:5e:66:19:f1:3c:53:aa:97:7e:09:61:4c:
          de:2e:99:d0:f9:ce:46:0b:b7:2b:0d:2c:e0:87:4d:
          91:81:90:f4:a0:33:f9:7d:0f:f2:b2:cb:5f:94:56:
          c2:6b:cf:2d:09:41:ea:4c:ab:84:82:da:fd:65:83:
          a5:9a:7d:6f:7c:a1:72:f3:0b:83:01:d6:85:ff:bd:
          27:f2:70:54:1e:f8:a5:00:2d:bc:e8:26:32:dd:20:
          87:28:3f:b1:ba:80:1e:de:09
        Exponent: 65537 (0x10001)
    X509v3 extensions:
      X509v3 Subject Key Identifier:
        36:81:43:FC:DB:22:EB:C7:D0:A0:BF:23:55:21:FC:EA:C6:71:AA:93
      X509v3 Authority Key Identifier:
        keyid:36:81:43:FC:DB:22:EB:C7:D0:A0:BF:23:55:21:FC:EA:C6:71:AA:93
```

```
        X509v3 Basic Constraints:
          CA:TRUE
    Signature Algorithm: sha1WithRSAEncryption
      27:0f:bf:44:1d:91:48:30:e4:3e:3f:25:0c:d6:f7:28:bd:d8:
      b7:75:16:b6:2a:14:be:6b:f9:5b:39:e7:a5:fa:7d:63:d3:13:
      1f:7b:92:81:45:87:68:df:96:72:87:87:5e:5d:83:6c:10:b0:
      a0:ec:b4:d5:8d:a2:ca:26:29:b4:ea:77:d8:f9:d8:2c:0a:1b:
      a6:23:07:c2:d1:4d:13:05:a3:35:fb:8d:cb:c4:40:fc:a9:26:
      a9:50:86:76:89:d3:71:49:25:a4:03:ff:12:6a:46:02:8d:5f:
      6e:19:92:84:52:a2:8c:99:5f:5f:75:c5:81:79:47:13:16:ef:
      1d:92
```

Example: **show ssl certificate** ***cert-name***

```
vxa22-6 (config) # show ssl certificate rsa1024.pem
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      a4:0e:e6:5f:10:d0:c0:79
    Signature Algorithm: sha1WithRSAEncryption
    Issuer: C=IN, ST=TamilNadu, L=Chennai, O=Ankeena, OU=Juniper,
CN=www.ankeena.com/emailAddress=tester@juniper.com
    Validity
      Not Before: Mar 26 12:03:09 2012 GMT
      Not After : Mar 26 12:03:09 2013 GMT
    Subject: C=IN, ST=TamilNadu, L=Chennai, O=Ankeena, OU=Juniper,
CN=www.ankeena.com/emailAddress=tester@juniper.com
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (1024 bit)
        Modulus (1024 bit):
          00:db:fc:ba:20:ae:5f:6c:41:9f:20:81:f7:4d:ca:
          02:a1:b8:16:21:59:e0:12:ef:2d:e3:6b:1d:48:2a:
          5c:cf:c2:eb:dd:fb:8a:93:59:d7:b1:b7:eb:22:2a:
          16:12:4c:65:34:6c:25:fb:e8:32:f3:f5:4c:ec:3d:
          1b:c7:94:5a:b2:9a:8d:a1:c6:e3:0a:cc:6d:8d:e1:
          89:c6:cd:57:b6:87:d1:e3:e5:ea:43:d5:2e:aa:dd:
          da:80:cc:2b:b8:3c:44:3c:ae:a6:a0:dd:1c:23:ed:
          d0:7b:ab:e0:97:16:c4:96:71:ff:67:9f:15:6a:f4:
          83:4f:31:48:60:6b:08:07:19
        Exponent: 65537 (0x10001)
    X509v3 extensions:
      X509v3 Subject Key Identifier:
        F4:20:68:B0:B7:ED:41:59:94:73:D1:BA:8B:47:A0:06:5F:A9:83:FC
      X509v3 Authority Key Identifier:
        keyid:F4:20:68:B0:B7:ED:41:59:94:73:D1:BA:8B:47:A0:06:5F:A9:83:FC

      X509v3 Basic Constraints:
        CA:TRUE
  Signature Algorithm: sha1WithRSAEncryption
    3f:01:05:2c:5e:1d:9f:e0:f1:02:ab:d9:d3:48:60:99:e4:2b:
    f5:54:4d:ce:b5:4f:57:ef:61:55:4e:cf:06:fa:d1:ab:72:dd:
    3e:f1:ab:a3:8b:4a:55:6c:d9:36:15:ee:cc:a2:5f:34:a4:a1:
    fc:b4:8b:5b:ba:65:27:33:1e:b8:4c:61:51:22:d6:07:93:5b:
    e8:a5:8d:66:77:29:29:12:fd:34:77:48:96:b5:db:f1:64:56:
```

```
d5:53:43:5c:a3:bc:eb:f4:9a:52:31:23:e2:80:43:49:07:04:
fb:11:7d:72:07:11:6d:2f:cc:c9:0e:e3:56:12:f8:9b:ab:b4:
cf:b9
```

Example: **show ssl cipher**

```
MFC1(config) # show ssl cipher
Cipher Configured   : eNULL
```

Related Documentation

- Content Ingest Manager Overview on page 171
- Setting Up the Content Ingest Manager on page 174
- Configuring HTTPS Fetch from Origin Servers on page 166.
- Viewing the Status of the HTTP Crawler on page 182
- Viewing HTTP Crawler Instance on page 182

Content Ingest Manager Expiration Management

- Content Ingest Manager Cache Object Expiration Management Overview on page 186
- Parent and Edge Crawlers Expire a Cached Object at the Same Time Overview on page 189
- Content Ingest Manager Crawler ASX File Expiration Overview on page 189
- Configuring the Parent and Edge Crawlers to Expire a Cached Object at the Same Time on page 190
- Configuring Content Ingest Manager Crawler ASX File Expiration on page 191

Content Ingest Manager Cache Object Expiration Management Overview

The Content Ingest Manager crawler configurable parameters allow you to configure the following expiration management features:

- Set objects cached by the parent and edge crawlers to expire at the same time, regardless of whether the parent and edge crawl start at the same time. See "Parent and Edge Crawlers Expire a Cached Object at the Same Time Overview" on page 189.
- Set the expiration time of Advanced Stream Redirector (ASX) files that store a playlist of WMV files. See "Content Ingest Manager Crawler ASX File Expiration Overview" on page 189.
- Set the cache age for cache-miss fetch objects at the namespace level.

Content Ingest Manager Crawler expiration management scenarios are as follows:

- **Scenario 1:** Parent and edge Media Flow Controllers populate content using Crawler:
  - Expiration of objects at the parent is based on the crawl refresh-interval at the parent.
  - Expiration of objects at the edge is based on the crawl refresh-interval at the edge.
  - For objects populated into the cache before the crawler completes downloading that object (such as objects populated using cache-miss fetch), the expiration is set

based on the headers from origin or the namespace, cache-age/cache-age-default configuration.

- **Scenario 2:** Parent Media Flow Controller populates content using Crawler and edge Media Flow Controllers populate content using cache-miss fetch:
  - Expiration of objects at the parent is based on the crawl refresh-interval at the parent.
  - Expiration of objects at the edge is based on the Cache-Control: max-age and Age headers inserted by the parent Media Flow Controller.
  - There may be a 1-second difference in the expiration of objects between the edge and the parent caches in this scenario. This is because of a few milliseconds difference in the clock even if both the edge and parent caches are synced up using NTP. For example, if an object is ingested at the parent cache at 17:05:00:000 (hours:mins:secs:milli secs) and served to the edge cache by 17:05:01:000, the age of the object in the parent would be 1 second. Edge cache would use the Age header along with the Cache-Control:max-age header to compute the expiry time of the object. If the clock in the edge cache is transitioning from 17:05:00:000 to

17:05:00:995, the expiry of the object at the edge may be less by 1 second when compared to the parent.

NOTE:

- If the Crawler is set to use the refresh-interval as the expiry, the expiry time for crawled content is always set based on an absolute time calculated based on the refresh interval. For example, if the crawl start time is 9:00 am and the refresh-interval is 10 minutes, the crawled content always expires at 9:10 am irrespective of whether the content was downloaded from the origin at 9:00 am or 9:05 am.
- If the refresh interval is set to 0 to enable one time crawl or if the Crawler is set to use origin-response based expiry, the expiry time is set based on:
  - Response headers from the origin and parent.
  - If the origin or parent did not send any cache expiry headers (such as Expires, Cache-Control: max-age, Age), the expiration of the object is set based on the namespace cache-age/cache-age-default configuration.
- Two methods to make objects cached at the parent and edge expire at the same time:
  - Configure the parent and edge crawler with the same start-time and refresh-interval.
  - Configure the edge to use origin-response to calculate expiry in the Crawler configuration instead of the refresh interval. In this configuration, the start-time and refresh-interval for the edge can be different from the parent.
- For all the crawled content, the existing Cache-Control headers are removed if present, and a new Cache-Control header with max-age is added based on the object expiration.
- The expiry time of an object that is revalidated during the crawl, is updated to the disk only if the remaining validity time of that object is greater than 30 seconds. This is done to use disk resources efficiently. There is no functional impact to the media delivery.

Related Documentation

- Parent and Edge Crawlers Expire a Cached Object at the Same Time Overview on page 189
- Configuring the Parent and Edge Crawlers to Expire a Cached Object at the Same Time on page 190
- Content Ingest Manager Crawler ASX File Expiration Overview on page 189
- Configuring Content Ingest Manager Crawler ASX File Expiration on page 191

Parent and Edge Crawlers Expire a Cached Object at the Same Time Overview

In Reverse Proxy deployments, you can configure Content Ingest Manager parent and edge crawlers to expire cached objects at the same time in the following scenarios:

- When you configure the parent and edge crawlers to have the same start time and refresh interval. (Objects must be populated in the parent and edge through a crawl.)
- When the parent crawler populates an object through crawl and the edge populates the object from the parent through cache-miss fetch.

Configure the crawler to set the expiry of the objects based on the origin-response using the **crawler *crawler_name* action set expiry origin-response** configure mode CLI command. This configuration must be done only at the edge Media Flow Controller, only when the edge and parent Media Flow Controllers don't crawl at the same time. See also the **crawler** command in the *Media Flow Controller CLI Command Reference*.

Related Documentation

- Content Ingest Manager Cache Object Expiration Management Overview on page 186
- Configuring the Parent and Edge Crawlers to Expire a Cached Object at the Same Time on page 190
- Content Ingest Manager Crawler ASX File Expiration Overview on page 189
- Configuring the Parent and Edge Crawlers to Expire a Cached Object at the Same Time on page 190
- Configuring Content Ingest Manager Crawler ASX File Expiration on page 191

Content Ingest Manager Crawler ASX File Expiration Overview

The Content Ingest Manager crawler generates an Advance Stream Redirector (ASX) file for each Windows Media Video (WMV) file found during a crawl. The ASX file contains a link to the video to be played. An ASX file is equivalent to a playlist or redirector file and is not published by the content provider. ASX file generation can be optionally configured under each crawler instance. All ASX files are generated and ingested before WMV files are preloaded.

You can configure the expiration time of Content Ingest Manager crawler ASX files. The default ASX file expiration time is 10 years, expressed in seconds. Configure the ASX expiration time on both the parent and edge Media Flow Controller crawlers only if you want to set a different ASX file expiration time.

ASX files are deleted when their corresponding WMV file is deleted as a result of a crawl. ASX files are not deleted when their corresponding WMV file is deleted as a result of a CLI command.

To configure ASX file expiration, use the **crawler *crawler_name* action auto-generate asx source-file wmv expiry *time_in_secs*** config mode CLI command. See also the **crawler** command in the *Media Flow Controller CLI Command Reference*.

Related Documentation

- Content Ingest Manager Cache Object Expiration Management Overview on page 186

- Parent and Edge Crawlers Expire a Cached Object at the Same Time Overview on page 189
- Configuring the Parent and Edge Crawlers to Expire a Cached Object at the Same Time on page 190
- Configuring Content Ingest Manager Crawler ASX File Expiration on page 191

Configuring the Parent and Edge Crawlers to Expire a Cached Object at the Same Time

Configure the Media Flow Controller edge crawler so that cached objects expire on both the edge and parent Media Flow Controllers at the same time regardless of the crawler start times. Configure the edge Media Flow Controller only if the edge and parent Media Flow Controllers do not crawl at the same time. You do not have to configure the parent Media Flow Controller crawler.

To configure the edge Media Flow Controller crawler cache object expiration:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. On the edge Media Flow Controller crawler, configure the expiry mechanism as origin-response.

   (config) # **crawler *crawler_name* action set expiry origin-response**

   Use the **crawler *crawler_name* no action set expiry origin-response** command to remove the expiry origin-response.

3. Show the crawler configuration.

   (config) # **show crawler *crawler_name***

   For example:

```
sh crawler <crawler name>
Crawler instance: test
Base URL: http://www.xyz.com/test/
Depth: 10
Refresh interval: 0
Status: inactive
 Preloaded file extensions:
  NONE
 Non-preloaded file extensions:
  NONE
Auto-generate: ASX from source-file WMV
Expiry Value for Auto Generation:   315360000
Expiry time based on the cache-control Enabled: yes
Scheduled start time: 1970/01/01 00:00:00
Scheduled stop time: 1970/01/01 00:00:00
Next crawl time: None
 Last Crawl Details:
start timestamp: None
```

```
end timestamp: None
Number of objects added: 0
Number of objects deleted: 0
Number of adds failed: 0
Number of deletes failed: 0
```

Related Documentation

- Content Ingest Manager Cache Object Expiration Management Overview on page 186
- Parent and Edge Crawlers Expire a Cached Object at the Same Time Overview on page 189
- Content Ingest Manager Crawler ASX File Expiration Overview on page 189
- Configuring Content Ingest Manager Crawler ASX File Expiration on page 191

Configuring Content Ingest Manager Crawler ASX File Expiration

Configure the ASX file expiration time on both the parent and edge Media Flow Controller crawlers only if you want to set a different expiration time.

To configure crawler ASX file expiration:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. On both the Media Flow Controller crawler, configure the ASX file expiration time, in seconds.

   crawler *crawler_name* action auto-generate asx source-file wmv expiry *time_in_secs*

   The default ASX file expiration time is 10 years, expressed in seconds (1 year equals 31, 556, 926 seconds. 2 years equals 63, 113, 851.9 seconds, and so forth).

3. Show the crawler configuration.

   show crawler *crawler name*

   For example:

```
sh crawler <crawler name>
Crawler instance: test
Base URL: http://www.xyz.com/test/
Depth: 10
Refresh interval: 0
Status: inactive
 Preloaded file extensions:
  NONE
 Non-preloaded file extensions:
  NONE
Auto-generate: ASX from source-file WMV
Expiry Value for Auto Generation:   315360000
Expiry time based on the cache-control Enabled: yes
Scheduled start time: 1970/01/01 00:00:00
Scheduled stop time: 1970/01/01 00:00:00
```

```
Next crawl time: None
  Last Crawl Details:
start timestamp: None
end timestamp: None
Number of objects added: 0
Number of objects deleted: 0
Number of adds failed: 0
Number of deletes failed: 0
```

Related Documentation

- Content Ingest Manager Cache Object Expiration Management Overview on page 186
- Parent and Edge Crawlers Expire a Cached Object at the Same Time Overview on page 189
- Configuring the Parent and Edge Crawlers to Expire a Cached Object at the Same Time on page 190
- Content Ingest Manager Crawler ASX File Expiration Overview on page 189

CHAPTER 7

Configuring Media Flow Controller Load Balancing

Media Flow Controller Load Balancing Overview

Load balancing allows you to distribute incoming requests across two or more Media Flow Controllers, providing scalability and high service availability.

Load balancers can operate in Layer 4 or Layer 7 mode when distributing load across Media Flow Controllers:

- Layer 4 load balancer—Requests are routed to the Media Flow Controller by a load balancer or a Layer 4 switch or router. When Media Flow Controller serves user requests, the responses go through the Layer 4 load balancer, and some performance impact can occur in this deployment mode.
- Layer 4 load balancer plus Direct Server Return (DSR)—This is one of the most popular modes as it allows deployment with an inexpensive load balancer. DSR allows the return data (response) to go directly from Media Flow Controller to the client. This

allows scaling to 10 Gbps with a relatively inexpensive load balancer. The load balancer itself need not scale to 10 Gbps. As long as it has enough bandwidth to handle the rate of incoming requests, this provides a good solution.

- Layer 7 load balancer—DSR does not work in this case. The load balancer must match the sum of the capacity of all the Media Flow Controllers to which the load balancer is load-balancing. However, rich Layer 7 policies based on URI, header, and so on, can be built on the load balancer, and traffic steering can be done in a more flexible way.

Media Flow Controller can be configured for DSR when load balancers distribute requests across caches in Layer 4 mode. Load balancing with DSR reduces traffic load on the load balancer because the response from the target server bypasses the load balancer and goes directly to the client. This chapter describes how Media Flow Controllers can be configured to work in Layer 4 or DSR mode.

Related Documentation

- Load Balancing with Direct Server Return on page 194
- Cluster Layer 7 Redirect on page 195
- BGP Traffic Steering Overview on page 196
- Provisioning Virtual IP Addresses Using Loopback Aliases Overview on page 203

Load Balancing with Direct Server Return

Load balancing with Direct Server Return (DSR) reduces traffic load on the load balancer because the response from the target server bypasses the load balancer and goes directly to the client.

To implement DSR in Media Flow Controller, the following configurations are required:

1. The server load balancer (SLB) and Media Flow Controller must be Layer 2 adjacent.
2. Media Flow Controller must have the destination load balancer virtual IP address (VIP) configured on a loopback or a network interface that will not broadcast that IP address on the network.
3. Media Flow Controller cannot use the address resolution protocol (ARP) to assign the VIP address its own media access control (MAC) address. You can disable ARP on a Media Flow Controller interface with **interface *interface_name* arp disable**.
4. The return response from Media Flow Controller must bypass the SLB.
5. Media Flow Controller responses are routed to a host that is not Layer 2 adjacent, using your configured route or gateway.

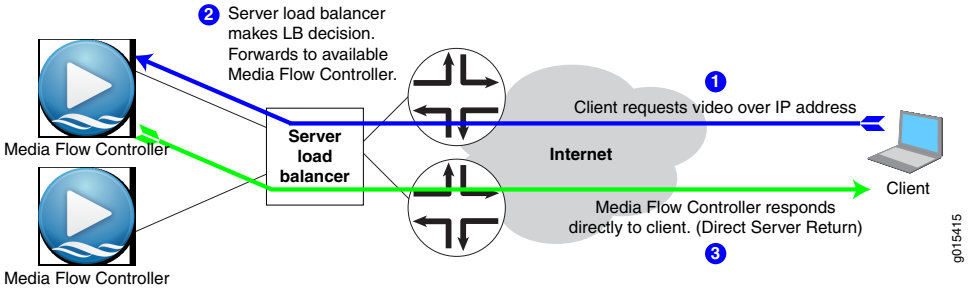
Examples:

- Layer 4 load balancer plus DSR—This is one of the most popular modes as it allows deployment with an inexpensive load balancer. DSR allows the return data (response) to go directly from Media Flow Controller to the client. This allows scaling to 10 Gbps with a relatively inexpensive load balancer. The load balancer itself need not scale to 10 Gbps. As long as it has enough bandwidth to handle the rate of incoming requests, this provides a good solution.

- Layer 7 load balancer—DSR does not work in this case. The load balancer must match the sum of the capacity of all the Media Flow Controllers to which the load balancer is load-balancing. However, rich Layer 7 policies based on URI, header, and so on, can be built on the load balancer, and traffic steering can be done in a more flexible way.

See Figure 8 on page 195, for an illustration.

Figure 8: Direct Server Return



Related Documentation

- Media Flow Controller Load Balancing Overview on page 193
- Cluster Layer 7 Redirect on page 195
- BGP Traffic Steering Overview on page 196
- Provisioning Virtual IP Addresses Using Loopback Aliases Overview on page 203

Cluster Layer 7 Redirect

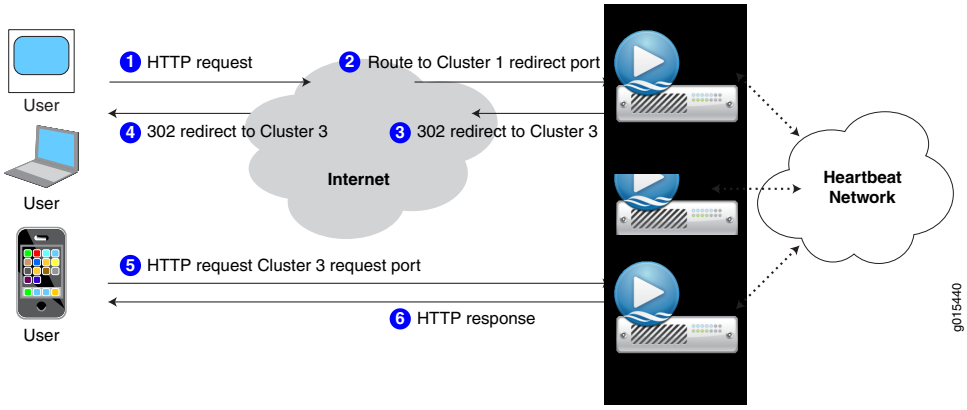
Layer 7 refers to the “application” layer, the seventh, in the Open Systems Interconnection (OSI) model. A Cluster Layer 7 Redirect is a way to load-balance a cluster of Media Flow Controllers using the Application transport layer.

NOTE: **Cluster Layer 7 Redirect is suited for deployments where Media Flow Controller is used for delivering large media objects such as videos or software installation packages.**

Each Media Flow Controller must provide distinct redirect and request HTTP listener IP address and port(s) for the Cluster Layer 7 traffic. The redirect IP address and port applies the node selection policy and initiates the HTTP 302 redirect to the target request IP address and port.

Cluster Layer 7 redirect capability is configured by defining a **cluster** object, which defines the Cluster Layer 7 attributes (for example, redirect and request IP and port), and attaching it to a **namespace**. The attached namespace can be an existing or new namespace. Cluster node status is determined using periodic HTTP "heartbeat" requests where the response contains status and node load data. See Figure 9 on page 196.

Figure 9: Cache Clustering Layer 7 Redirect



Related Documentation

- Media Flow Controller Load Balancing Overview on page 193
- Load Balancing with Direct Server Return on page 194
- BGP Traffic Steering Overview on page 196
- Provisioning Virtual IP Addresses Using Loopback Aliases Overview on page 203

BGP Traffic Steering Overview

Border Gateway Protocol (BGP) Traffic Steering allows you to direct traffic from a peering router, using certain destination IP addresses, that correspond to origin servers.

In current Media Flow Controller Transparent Proxy deployments, peering routers are configured using a Policy-based Forwarding (PBF) filter to forward all HTTP traffic on TCP port 80 to Media Flow Controller. The Media Flow Controller listens for HTTP requests and either serves the content from its local cache or gets the content from origin servers. The problem with the current deployment is Media Flow Controller is exposed to a lot of non-cacheable traffic that wastes valuable caching resources. The solution is to separate the cacheable and non-cacheable traffic to provide better bandwidth savings and throughput by efficiently using Media Flow Controller resources.

BGP traffic steering allows Media Flow Controller to only receive cacheable traffic by advertising certain IP addresses through BGP to the peering router. Destination IP selection is performed using access log analysis at the customer premise. The configuration of the destination IP addresses, using the CLI, is static as opposed to a dynamic mechanism where IP addresses are added and deleted using intelligent traffic inspection.

Juniper Networks granularly examines the access logs that are generated by a Media Flow Controller to determine which origin servers to add to this list of destination IP addresses and to get the best possible bandwidth savings. Juniper Networks also provides access log generation software that generates access logs for analysis.

Media Flow Controller semi-transparent proxy deployment mode is supported when you configure BGP traffic steering. This is not a limitation of Media Flow Controller, but inherent to the BGP feature implementation. When using BGP traffic steering, the Media Flow

Controller IP address is used to fetch cache-miss traffic from origin servers. The maximum number of routes that Media Flow Controller can advertised is 10,000.

Two BGP traffic steering use case configurations are provided:

- Advertising a Media Flow Controller Interface Network to Neighbors on page 197
- Advertising the Media Flow Controller Outside Network To Neighbors on page 198

Related Documentation

- Advertising a Media Flow Controller Interface Network to Neighbors on page 197
- Advertising the Media Flow Controller Outside Network To Neighbors on page 198
- Configuring BGP Traffic Steering Optional Router ID, Neighbor Logging, Shutdown, and Timer Settings on page 199
- Verifying the BGP Traffic Steering Configuration and Operation on page 200
- Clearing Media Flow Controller BGP Traffic Steering on page 202

Advertising a Media Flow Controller Interface Network to Neighbors

You can configure Media Flow Controller to communicate with a BGP peer using a neighbor statement and advertise a route to that neighbor.

Optionally, you can override the router ID, configure the BGP keepalive and hold time.

To advertise Media Flow Controller interface networks to neighbors:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Configure the Media Flow Controller BGP neighbors.

   # **router bgp** *local-as* **neighbor** *neighbor ip* **remote-as** *remote-as*

   For example, enter **router bgp 64512 neighbor 1.1.1.11 remote-as 64512**

   The BGP configurable parameters include:

   - **local-as**— local autonomous system number (ASN) is between 1-4294967295. The block of ASNs from 64,512 through 65,534 are designated for private use. ASN 23,456 is reserved for use in an ASN pool transition. Per RFC 4893, routers should accept 4 octet ASNs.
   - **neighbor ip**—A neighbor IP address. You can configure multiple neighbors.
   - **remote-as**—A remote autonomous system (AS) number.

3. Configure Media Flow Controller to update a neighbor about directly connected networks.

   # **router bgp** *as-number***redistribute connected**

   For example, # **router bgp 64512 redistribute connected**.

4. (Optional) Override the router ID and manually configure a different one.

   See "Configuring BGP Traffic Steering Optional Router ID, Neighbor Logging, Shutdown, and Timer Settings" on page 199.

5. (Optional) Configure keep alive interval and hold time timers in seconds.

   See "Configuring BGP Traffic Steering Optional Router ID, Neighbor Logging, Shutdown, and Timer Settings" on page 199.

Related Documentation

- BGP Traffic Steering Overview on page 196
- Advertising the Media Flow Controller Outside Network To Neighbors on page 198
- Configuring BGP Traffic Steering Optional Router ID, Neighbor Logging, Shutdown, and Timer Settings on page 199
- Verifying the BGP Traffic Steering Configuration and Operation on page 200
- Clearing Media Flow Controller BGP Traffic Steering on page 202

Advertising the Media Flow Controller Outside Network To Neighbors

You can configure Media Flow Controller to advertise outside networks to neighbors. You advertise to BGP neighbors which IP network you are advertising to them. That way when Media Flow Controller sends traffic to them, they know how to respond back.

To configure Media Flow Controller to advertise outside networks to neighbors:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Configure BGP settings.

   Configure an AS number between 1-4294967295.

   # **router bgp** ***local-as*** **neighbor** ***neighbor ip*** **remote-as** ***remote-as***

   For example, enter **router bgp 64512 neighbor 1.1.1.11 remote-as 64512**.

3. Specify a network to announce using BGP.

   The mask length is between 0 and 32.

   # **router bgp** ***as-number*****network** ***prefix/mask length***

   For example, enter **router bgp 64512 network 10.10.10.0 /24**

4. (Optional) Override the router ID and manually configure a different one.

   See "Configuring BGP Traffic Steering Optional Router ID, Neighbor Logging, Shutdown, and Timer Settings" on page 199.

5. (Optional) Configure keep alive interval and hold time timers.

See "Configuring BGP Traffic Steering Optional Router ID, Neighbor Logging, Shutdown, and Timer Settings" on page 199.

Related Documentation

- BGP Traffic Steering Overview on page 196
- Advertising a Media Flow Controller Interface Network to Neighbors on page 197
- Configuring BGP Traffic Steering Optional Router ID, Neighbor Logging, Shutdown, and Timer Settings on page 199
- Verifying the BGP Traffic Steering Configuration and Operation on page 200
- Clearing Media Flow Controller BGP Traffic Steering on page 202

Configuring BGP Traffic Steering Optional Router ID, Neighbor Logging, Shutdown, and Timer Settings

When configuring BGP traffic steering required settings, such as the AS number and neighbor IP address, you can also configure optional settings, such as override the router ID and set a keep alive interval and hold time. You can configure the device to loge messages about neighbors going up or down and shut down the neighbor.

1. Manually Overriding the BGP Router ID on page 199
2. Logging Neighbor State Messages on page 199
3. Shutting Down Neighbors on page 200
4. Configuring the BGP Traffic Steering Keepalive Interval and Hold Time on page 200

Manually Overriding the BGP Router ID

To manually override the BGP router ID:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. (Optional) Override the router ID and manually configure a different one.

   The router ID is using IP address format.

   # **router bgp** ***as-number*** **bgp router-id** ***router-id***

   For example, enter **router bgp 64512 bgp router-id 1.1.1.1**.

Logging Neighbor State Messages

To manually enable the neighbor state logging messages:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. (Optional) Set the device to log neighbor state-change messages at the notice level.

   This feature is enabled by default.

   # **router bgp *as-number* bgp log-neighbor-changes**

   For example, enter **router bgp 64512 bgp log-neighbor-change**.

Shutting Down Neighbors

To manually shut down a BGP neighbor:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. (Optional) Manually shut down the neighbor relationship with the specified BGP neighbor.

   # **router bgp *as-number* neighbor *ip_address* shutdown**

   For example, enter **router bgp 64512 neighbor 1.1.1.1 shutdown**.

Configuring the BGP Traffic Steering Keepalive Interval and Hold Time

You can set the BGP keepalive interval and hold time. BGP systems exchange keepalive messages to determine whether a link or host has failed or is no longer available. The keepalive interval is the interval between 2 keepalive messages. The hold time is the maximum number of seconds allowed to elapse between successive keepalive or update messages that BGP receives from a peer.

To configure the BGP timers:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. (Optional) Configure the BGP keepalive interval and hold time, in seconds.

   The hold time must be at least 3 times the keepalive interval.

   # **router bgp *as-number* timers bgp *keep-alive interval hold-time***

   For example, enter **router bgp 64512 timers bgp 60 180**.

Verifying the BGP Traffic Steering Configuration and Operation

1. Verifying the BGP Traffic Steering Configuration on page 201
2. Verify BGP Traffic Steering Operation on page 201

Verifying the BGP Traffic Steering Configuration

Using the **show run** CLI command, make sure all BGP commands that you configured are reflected in the show output.

To verify the BGP traffic steering configuration:

1. Enter the **show run** CLI command.

```
...
##
## BGP Configuration
##
  router bgp 64512 bgp router-id 1.1.1.1
  router bgp 64512 neighbor 1.1.1.11 remote-as 64512
  router bgp 64512 network 10.10.10.0 /24
  router bgp 64512 redistribute connected
  router bgp 64512 timers bgp 60 180
```

2. Check to see that your BGP traffic steering configuration is present in the **show run** CLI command output.

Verify BGP Traffic Steering Operation

When BGP is successfully running, you should see the BGP neighbor remote router ID, and the state should be **Established**.

If BGP is not running, the router will not send traffic to Media Flow Controller, and no caching occurs.

The **show bgp** CLI command displays all the configured BGP routes and the current state of the BGP session, including counters.

To verify the BGP traffic steering operation:

1. Enter the show bgp CLI command:

   > **show bgp**

   The arrows (—>) indicate whether neighbor relationship is successfully established in the **show bgp** CLI command output.

```
BGP local AS number is 123

BGP neighbor is 10.157.34.104, remote AS 123, local AS 123, internal link
—> BGP version 4, remote router ID 1.1.1.1
—> BGP state = Established, up for 00:00:19
Last read 03:14:26, hold time is 180, keepalive interval is 60 seconds
 Neighbor capabilities:
  4 Byte AS: advertised and received
  Route refresh: advertised and received(old & new)
  Address family IPv4 Unicast: advertised and received
 Message statistics:
  Inq depth is 0
  Outq depth is 0
```

```
                    Sent     Rcvd
  Opens:            2        0
  Notifications:    0        0
  Updates:          1        2
  Keepalives:       2        1
  Route Refresh:    0        0
  Capability:       0        0
  Total:            5        3
 Minimum time between advertisement runs is 5 seconds

For address family: IPv4 Unicast
 Community attribute sent to this neighbor(both)
 3 accepted prefixes

 Connections established 1; dropped 0
  Last reset never
Local host: 10.157.43.138, Local port: 179
Foreign host: 10.157.34.104, Foreign port: 40974
Nexthop: 10.157.43.138
Nexthop global: fe80::225:90ff:fe1a:9132
Nexthop local: ::
BGP connection: non shared network
Read thread: on  Write thread: off

—>BGP table version is 0, local router ID is 10.157.43.138
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
         r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network        Next Hop          Metric LocPrf Weight Path
*>i1.2.3.0/24     10.157.34.104          0   100     0 i
*>i10.2.3.0/24    10.157.34.104          0   100     0 i
*> 10.157.32.0/19  0.0.0.0               1       32768 ?
* i              10.157.34.104         1   100     0 ?

 Total number of prefixes 3
```

Related Documentation

- BGP Traffic Steering Overview on page 196
- Advertising a Media Flow Controller Interface Network to Neighbors on page 197
- Advertising the Media Flow Controller Outside Network To Neighbors on page 198
- Configuring BGP Traffic Steering Optional Router ID, Neighbor Logging, Shutdown, and Timer Settings on page 199
- Clearing Media Flow Controller BGP Traffic Steering on page 202

Clearing Media Flow Controller BGP Traffic Steering

When you clear the BGP traffic steering configuration, Media Flow Controller reestablishes neighbor relationships.

To clear a Media Flow Controller BGP relationship:

1. Enter the CLI configure mode.

> **enable**

 # configure terminal

2. Clear the BGP relationship.

 # clear bgp

Related Documentation

- BGP Traffic Steering Overview on page 196
- Advertising a Media Flow Controller Interface Network to Neighbors on page 197
- Advertising the Media Flow Controller Outside Network To Neighbors on page 198
- Configuring BGP Traffic Steering Optional Router ID, Neighbor Logging, Shutdown, and Timer Settings on page 199
- Verifying the BGP Traffic Steering Configuration and Operation on page 200

Provisioning Virtual IP Addresses Using Loopback Aliases Overview

Virtual IP (VIP) address provisioning eliminates the service-level dependency upon network interfaces, thus avoiding service disruption due to restart and bringing in IP-based access restriction per namespace.

The Media Flow Controller listens on up to 128 network interfaces. This includes physical interfaces (such as eth0), and logical interfaces (such as bond, aliases, and loopback). If you make any changes to a network interface configuration, such as creating a new interface, deleting an interface, or enabling HTTP for an interface, the delivery service must be restarted so that the changes take effect.

NOTE: **For HTTP-based media delivery, the no service restart CLI command is required. For HTTPS delivery, you must restart the mod-ssl service when you configure an alias IP address under the loopback interface.**

When you configure a virtual IP address using loopback aliases, Media Flow Controller automatically listens on newly configured loopback VIP addresses and receives HTTP requests for the new VIP address on any physical or logical interface without a delivery service restart. Thus, VIP address provisioning using loopback aliases allows interface connection failover from one interface to another and minimizes service outages. All delivery interfaces are typically bonded together. This is applicable to all network interfaces inclusive of physical, logical (bond), and aliases. Also only for IPv4 addresses, the **[no] delivery protocol http interface *if_nam*** command also needs no delivery engine restart, if the interface is IPv4 only.

You can configure a namespace to only accept connections from a configured list of VIP addresses. Use the **[no] namespace *name*bind ip *ip*** command to specify the valid IP addresses for which the namespace will accept connections. You can associate up to eight VIP addresses. The default value is None, in which the namespace accepts connections from all VIP addresses.

The following service-level parameters of an interface through which a connection is received are used for bit-rate computations:

- Maximum bandwidth supported on an interface
- Total number of sessions on the interface
- Interface credit

Media Flow Controller reports VIP address statistics, IP address, index name, and total sessions at an instance of time. It generates the following monitoring and statistics counters:

- VIP address in integer
- Total sessions currently handled on this VIP address
- Total alias (VIPs) interfaces configured

To configure a VIP address using loopback aliases, use the **interface lo alias *alias index* ip *ip-address netmask*** configure mode CLI command.

Related Documentation

- Configuring Provisioning Virtual IP Addresses Using Loopback Aliases on page 204
- Example VIP Address Provisioning Configuration on page 205
- Binding Provisioned Virtual IP Addresses to a Namespace on page 206
- Virtual IP Address Provisioning Counters on page 207

Configuring Provisioning Virtual IP Addresses Using Loopback Aliases

The Media Flow Controller listens on up to 128 network interfaces, including physical interfaces (such as eth0), and logical interfaces (such as bond, aliases, and loopback). When you configure virtual IP addresses (VIPs) using loopback aliases, Media Flow Controller automatically listens on newly configured loopback VIP addresses and receives HTTP requests for the new VIP address on any physical or logical interface without a delivery service restart.

To provision a VIP address using a loopback interface alias:

1. Enter CLI configuration mode.

   > **enable**

   # **configure terminal**

2. Add a new VIP address and subnet mask for the loopback interface.

   # **interface lo alias <alias index> ip *ip address netmask***

   For example, interface lo alias 3 ip address 3.1.1.1 255.255.255.0.

   Enter a netmask either as a dotted quad (for example, 255.255.255.0) or as a mask length after a slash (for example, /29). The mask length must be between 1 and 32, inclusive (or 0 if the address is also 0).

Send traffic to verify that Media Flow Controller listens on the VIP address interface.

To delete a VIP address:

1. Enter CLI configuration mode.

   > **enable**

   # **configure terminal**

2. Delete a VIP address.

   (config) # **no interface lo alias *alias index***

To shut down or turn on VIP address provisioning:

1. Enter CLI configuration mode.

   > **enable**

   # **configure terminal**

2. Shut down VIP address provisioning.

   (config) # **interface lo: *alias index* shutdown**

3. Turn on VIP address provisioning.

   (config) # **no interface lo: *alias index***

Related Documentation

- Provisioning Virtual IP Addresses Using Loopback Aliases Overview on page 203
- Example VIP Address Provisioning Configuration on page 205
- Binding Provisioned Virtual IP Addresses to a Namespace on page 206
- Virtual IP Address Provisioning Counters on page 207

Example VIP Address Provisioning Configuration

The following sample environment shows a system with a VIP address and one physical connection.

```
vxa (config) # interface lo alias 1 ip address 20.20.20.20 255.255.255.0
vxa (config) # sh interfaces
Interface eth0 state
  Admin up:        yes
  Link up:        yes
  IP address:      10.157.43.58
  Netmask:         255.255.224.0
<output truncated>
Interface lo state
  Admin up:        yes
  Link up:        yes
  IP address:      127.0.0.1
  Netmask:         255.0.0.0
```

```
    Secondary address:  20.20.20.20/24 (alias: 'lo:1')
<output truncated>
```

Related Documentation

- Provisioning Virtual IP Addresses Using Loopback Aliases Overview on page 203
- Configuring Provisioning Virtual IP Addresses Using Loopback Aliases on page 204
- Binding Provisioned Virtual IP Addresses to a Namespace on page 206
- Virtual IP Address Provisioning Counters on page 207

Binding Provisioned Virtual IP Addresses to a Namespace

Binding virtual IP addresses using a loopback alias to a namespace allows the administrator to configure Media Flow Controller to listen on new interfaces without having to configure the golden configuration.

Media Flow Controller can have one or more alias IP addresses assigned for receiving HTTP requests from users. The alias IP address are generally provided to the subscribers by the content router or by a DNS server:

- These are virtual IP addresses used by the load balancer when the Media Flow Controllers are behind a local Layer 4 to Layer 7 load-balancing server.
- These are the IP addresses used by the router to receive subscriber traffic when the Media Flow Controller is behind a router using Equal Cost Multipath (ECMP) routing for Layer 3 load balancing.
- It is the IP address used by the Media Flow Controller to receive subscriber traffic when there is only one Media Flow Controller in a location that is serving traffic.

In all three cases, these alias IP addresses are different than the one which is the unique (per Media Flow Controller) primary public IP address used by the content router to request load feedback information.

By binding the alias IP to a namespace using the **bind ip** command, a specific user's connection is restricted from fetching content from other namespaces in the Media Flow Controller.

To bind a new VIP address to a namespace:

1. Enter CLI configuration mode.

   > **enable**

   # **configure terminal**

2. Bind an IP address to a namespace.

   The namespace *name* bind ip ? CLI Help text lists the available IP addresses.

   (config) # **namespace *name* bind ip *ipipv4***

   For example, namespace *test* bind ip *127.0.0.2*.

3. Optionally, unbind an IP address from a namespace.

(config) # **no namespace** *name* **bind ip** *ipipv4*

4. Show the IP address(es) bound to the namespace.

   show namespace *name*

```
...
 Bound IP Address:
  Default (All IP's)

     -OR-

 Bound IP Address:
  10.157.43.209
  127.0.0.1
```

Related Documentation

- Provisioning Virtual IP Addresses Using Loopback Aliases Overview on page 203
- Example VIP Address Provisioning Configuration on page 205
- Configuring Provisioning Virtual IP Addresses Using Loopback Aliases on page 204
- Virtual IP Address Provisioning Counters on page 207

Virtual IP Address Provisioning Counters

Media Flow Controller reports virtual IP statistics, IP address, index name, and total sessions at an instance of time. It generates the following monitoring and statistics counters:

- VIP address in integer
- Total sessions currently handled on this VIP address
- Total alias (VIP) interfaces configured

Table 20 on page 207 shows the VIP address provisioning counter names and descriptions.

Table 20: VIP Address Provisioning Counters

Counter name	Description
net_port. <VIP_IF_NAME>.vip_addr	VIP address in integer.
net_port. <VIP_IF_NAME>.tot_sessions	Total sessions currently handled on this VIP address.
glob_tot_alias_interface	Total alias (VIPs) interfaces configured.

Related Documentation

- Provisioning Virtual IP Addresses Using Loopback Aliases Overview on page 203
- Configuring Provisioning Virtual IP Addresses Using Loopback Aliases on page 204
- Example VIP Address Provisioning Configuration on page 205
- Binding Provisioned Virtual IP Addresses to a Namespace on page 206

CHAPTER 8

Clustering Media Flow Controllers

Clustering Overview

Clustering involves grouping separate cluster nodes so that they act like a single cluster node. Members of a cluster are usually physically and topographically colocated.

Media Flow Controllers can be clustered to increase the effective size of the cache using a proxy cluster, or an origin cluster. Using proxy clustering, origin clustering, or both, has the advantage of significantly increasing the degree of reliability, availability, and scalability of network operations than if only a single Media Flow Controller or origin server is used. In all cases, origin servers can be either web servers, or Media Flow Controllers.

Use proxy clustering when you want to eliminate redirects by using proxy to move content from the target node to deliver smaller sized content, such as small documents.

There are several types of origin clustering:

- Consistent hash—Used to distribute HTTP requests across a set of origin servers.
- Origin failover and escalation—Deployed when you want Media Flow Controller to address a collection of origin servers in a specific order with consideration for configurable HTTP response codes denoting escalation to the next node.
- Cluster weighted round-robin—Deployed when you want Media Flow Controller to address a collection of origin servers with varying capabilities denoted by a configurable weight.

Related Documentation

- Consistent Hash for Mapping Requests to Servers on page 210
- Origin Failover and Escalation on page 212
- Cluster Weighted Round-Robin Load Distribution on page 213
- Configuring Origin Cluster Weighted Round-Robin on page 216
- Cluster Stacking on page 217
- Transparent Cluster Overview on page 218
- Configuring Transparent Clustering on page 220
- Media Flow Controller Server Map Overview on page 223

Consistent Hash for Mapping Requests to Servers

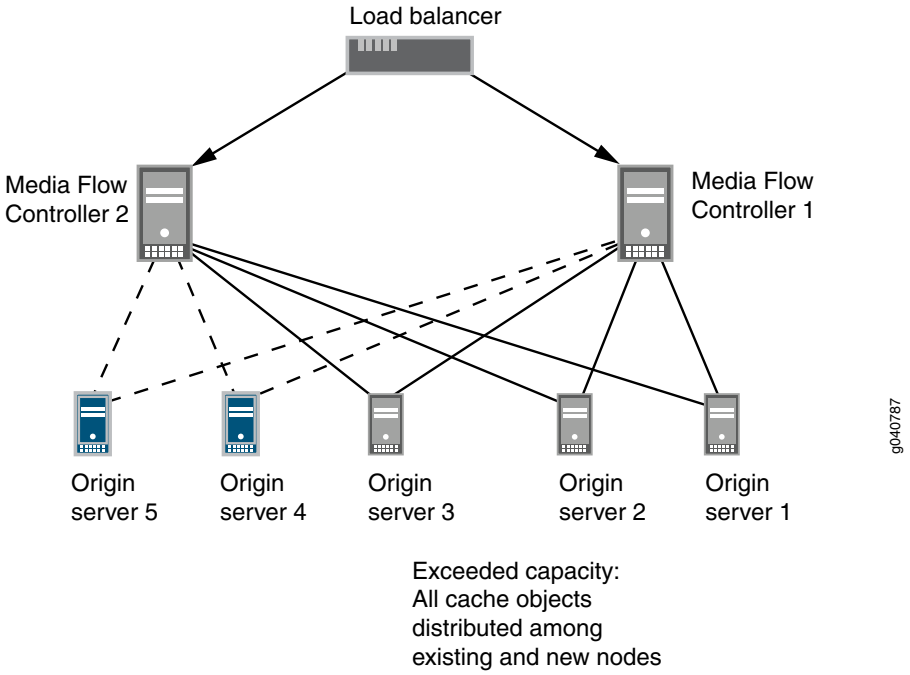
Consistent hash is the most basic form of clustering available. It is a computational mapping technique for distributing requests among a changing group of nodes in a cluster without the need for caches to communicate with one another. Each node in the cluster independently computes the object distribution of the cache using the same mapping function.

Because the strategy behind the consistent hash algorithm is to hash objects using the same computational hash function, any requests targeted to any of the nodes see exactly the same object distribution on that cache. This maps objects to the origin servers in the same way every time, so that when another origin server node is added, it takes its uniform share of objects from all other origin servers.

In this way, the consistent hash technique allows the expansion of the cluster by redistributing objects uniformly among additional nodes when more origin server resources are needed. Again, processing overhead is low because the entire hash process does not require any queries, broadcasts, or other forms of communication between origin servers making up the cluster.

In a clustering configuration, when an origin server node is removed or goes down, its objects are uniformly redistributed among the remaining origin server nodes. Cluster membership is determined using a periodic heartbeat to cluster members. When additional origin servers are added to expand the origin server resources, all cache objects are uniformly redistributed among the new and existing origin server resources, as in the example shown in Figure 10 on page 211.

Figure 10: Example of an Origin Escalation Cluster



When the capacity of Origin server 3 is exceeded and the administrator decides to expand its capacity by adding two new Origin servers 4 and 5, the consistent hash redistributes all cache objects evenly among the existing and new origin servers.

The advantages of consistent hashing over other hashing are:

- No communication is required between nodes. This reduces transactional latency because overhead operations, such as queries, are avoided.
- Because of the hashing algorithm, nodes correctly compute the same views of the cache content and can identify the target node using this local computation.
- Objects are distributed uniformly among the nodes.
- Objects are sticky after cluster node reconfiguration:
  - When a node is deleted, existing entries mapped to the same node and objects associated with the deleted node are uniformly distributed among the remaining nodes.
  - When a preexisting node is restored, entries map to the same nodes as before node deletion.
  - When a new node is added, an equal portion of the address space is remapped to the new node, resulting in a uniform distribution among all nodes.

Related Documentation

- Clustering Overview on page 209
- Origin Failover and Escalation on page 212
- Cluster Weighted Round-Robin Load Distribution on page 213

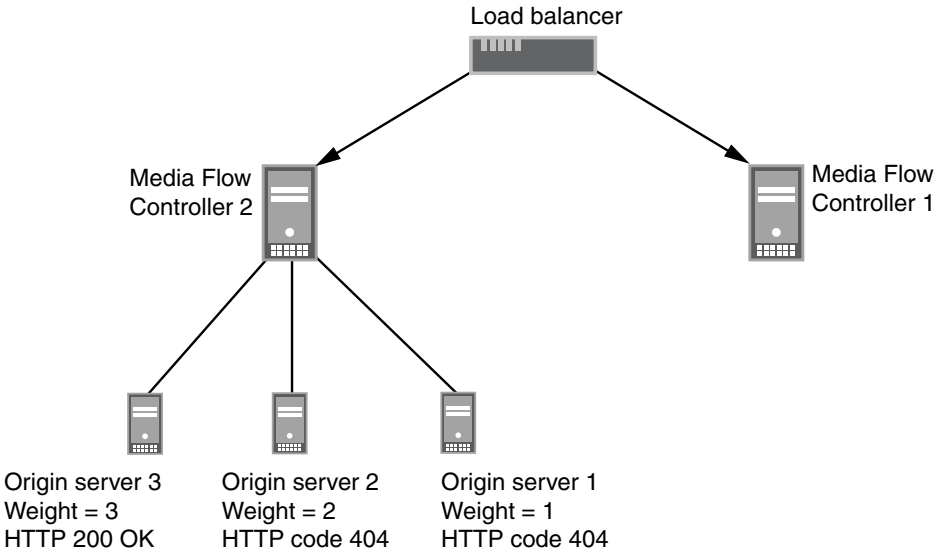
Origin Failover and Escalation

Use Origin failover and escalation clustering when you want Media Flow Controller to address a collection of origin servers in a specific order with consideration for configurable HTTP response codes denoting escalation to the next node. When origin escalation is configured, any number of online nodes is logically viewed as one node. The Media Flow Controller clustering heartbeat mechanism tracks the status of each node to determine which nodes in a cluster are active.

Requests are sent to nodes in a specific order, depending upon their preassigned weights until either the request is satisfied or an HTTP configured response code denoting escalation routes the request to the next weighted node. If the next node also responds with an HTTP configured response code denoting escalation, the next weighted node in the hierarchy is tried, and so on until all known available nodes in the hierarchy have been tried.

As shown Figure 11 on page 213, when Media Flow Controller 2 sends a request to Origin server 1 with an assigned hierarchical weighting of 1, an HTTP 404 response is returned (not found, configured response code denoting escalation), and a second request is sent to Origin server 2. When Origin server 2 is assigned a weighting of 2, it also returns an HTTP 404 response (configured response code denoting escalation), then a third request is sent to Origin server 3, where the request is resolved.

Figure 11: Example of an Origin Cluster Escalation



Related Documentation

- Clustering Overview on page 209
- Consistent Hash for Mapping Requests to Servers on page 210
- Cluster Weighted Round-Robin Load Distribution on page 213
- Configuring Origin Cluster Weighted Round-Robin on page 216
- Cluster Stacking on page 217
- Transparent Cluster Overview on page 218
- Configuring Transparent Clustering on page 220
- Media Flow Controller Server Map Overview on page 223

Cluster Weighted Round-Robin Load Distribution

When Media Flow Controller origin servers have varying CPU processing power or disk storage capabilities, it is advantageous to balance the load by applying a bias among them to more effectively utilize their capabilities. Use cluster weighted round-robin when you want Media Flow Controller to address a collection of origin servers with varying capabilities denoted by a configurable weight.

In a simple round-robin configuration, each Media Flow Controller request is distributed sequentially in a circular fashion around the pool of origin servers. Using this method, all the origin servers are treated as equals with a round-robin weight of 1, without regard to the node's processing capabilities.

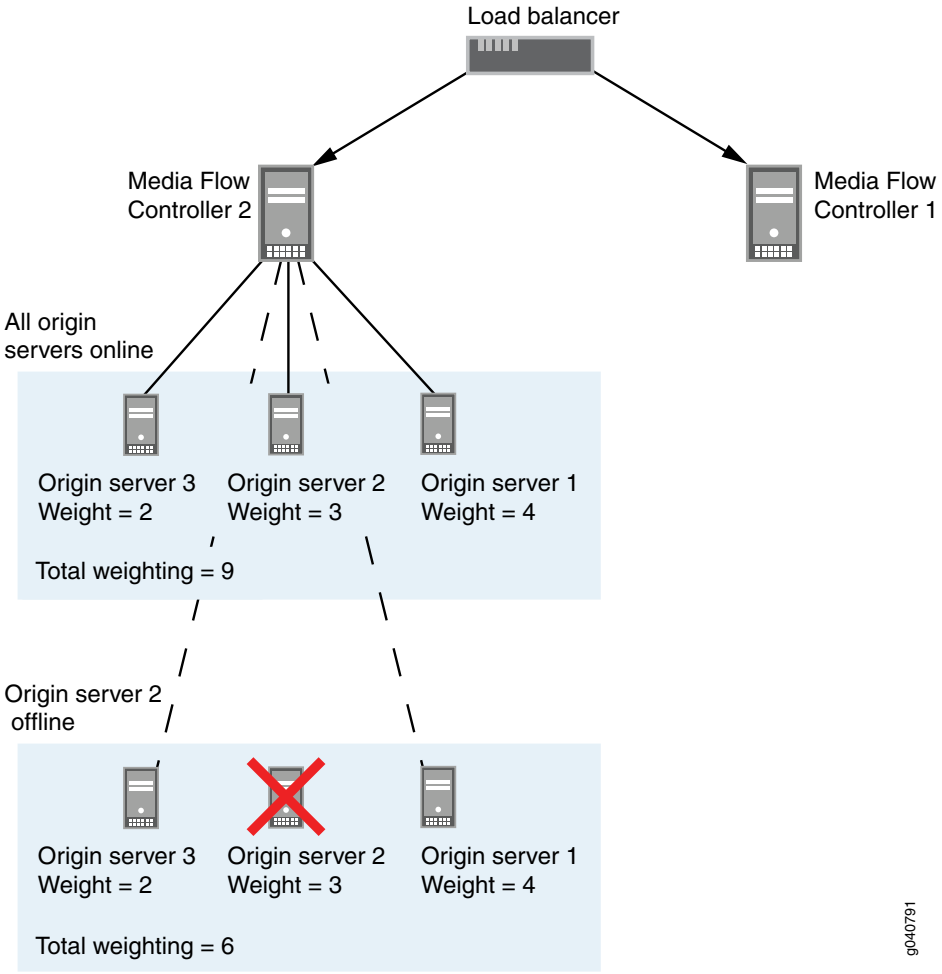
In a weighted round-robin configuration, even though each request is distributed sequentially in a circular fashion around the pool of origin servers, a bias is assigned to the nodes so that more requests are sent to those nodes with greater CPU processing

power or disk storage capacity. Origin server capabilities are predetermined by the network administrator and assigned a configurable weight factor in the range of 1 through 8, allowing the administrator to balance the request load among the origin server nodes based on their processing and storage capabilities. Generally, the higher the weight, the higher the numerical value assigned, where a weight of 8 is associated with origin servers with the most powerful processing capabilities. Weighted round-robin configuration is advantageous when there are significant differences in the capabilities of the origin servers in the pool.

As Figure 12 on page 215 illustrates, Origin server 1 is assigned the highest weight (4) because it has the highest processing and storage capabilities, Origin server 2 is assigned the second highest weight (3), and Origin server 3 is assigned the lowest weight (2). When all three origin servers are functioning normally, the overall load distribution weighting is 9 (4+3+2), so that four requests are sent to Origin server 1, three requests are sent to Origin server 2, and two requests are sent to Origin server 3.

When Origin server 2 goes offline, as shown in the figure, the heartbeat network acknowledges that Origin server 2 is no longer online, resulting in a new overall distribution weighting of 6. The load is then redistributed between Origin servers 1 and 3, so that four requests are now sent to Origin server 1 and two requests are sent to Origin server 3.

Figure 12: Example of Weighted Round-Robin Load Balancing



Related Documentation

- Clustering Overview on page 209
- Consistent Hash for Mapping Requests to Servers on page 210
- Origin Failover and Escalation on page 212
- Configuring Origin Cluster Weighted Round-Robin on page 216
- Cluster Stacking on page 217
- Transparent Cluster Overview on page 218
- Configuring Transparent Clustering on page 220
- Media Flow Controller Server Map Overview on page 223

Configuring Origin Cluster Weighted Round-Robin

To configure origin cluster weighted round-robin, first create a origin round-robin map format type of server map to send requests to the origin using the round-robin mode. Then create a namespace and select the origin round-robin server map. The server map uses the origin round-robin map XML file to set the heartbeat path, round-robin weight, and other details of each origin server.

Before you begin, create an origin round-robin map XML file as shown in the following example. See the "Media Flow Controller Server Map Overview" on page 223 for a complete explanation of server maps and their uses. You can copy this XML file into an editor and configure it to suit your specific needs. To create this file, you need to know:

- FQDN or IP address of the origin server.
- TCP port number of the origin server.
- Heartbeat path, which is the URI to the heart beat node.
- Round-robin weight (1 to 8) of each origin server.

Example of origin round-robin map XML file content:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE OriginRoundRobinMap SYSTEM "OriginRoundRobinMap.dtd">
 <OriginRoundRobinMap>
 <Header>
   <Version>1.0</Version>
   <Application>MapXML</Application>
 </Header>
 <OriginRoundRobinMapEntry>
   <Origin>172.19.172.84</Origin>
   <Port>80</Port>
   <Options>heartbeatpath=/heartbeat.html,round_robin_weight=1</Options>
 </OriginRoundRobinMapEntry>
 <OriginRoundRobinMapEntry>
   <Origin>xxx.com</Origin>
   <Port>80</Port>
   <Options>heartbeatpath=/heartbeat.html,round_robin_weight=1</Options>
 </OriginRoundRobinMapEntry>
 </OriginRoundRobinMap>
```

See "Media Flow Controller Server Map Overview" on page 223 for more information about server maps, and their uses.

To configure a server map, select the origin round-robin map, and an origin-server namespace, complete the following steps:

1. Define an origin round-robin map format type server-map reference that contains details about the HTTP origin servers and sends requests to the origin servers using the round-robin mode.

   server-map *name* format-type origin-round-robin-map

2. Configure a namespace, specifying which HTTP origin server to contact when the Media Flow Controller is unable to serve the object from cache and reference a server map.

   namespace *name* origin-server http server-map *server-map-name*

Related Documentation

- Clustering Overview on page 209
- Consistent Hash for Mapping Requests to Servers on page 210
- Origin Failover and Escalation on page 212
- Cluster Weighted Round-Robin Load Distribution on page 213
- Cluster Stacking on page 217
- Transparent Cluster Overview on page 218
- Configuring Transparent Clustering on page 220
- Media Flow Controller Server Map Overview on page 223

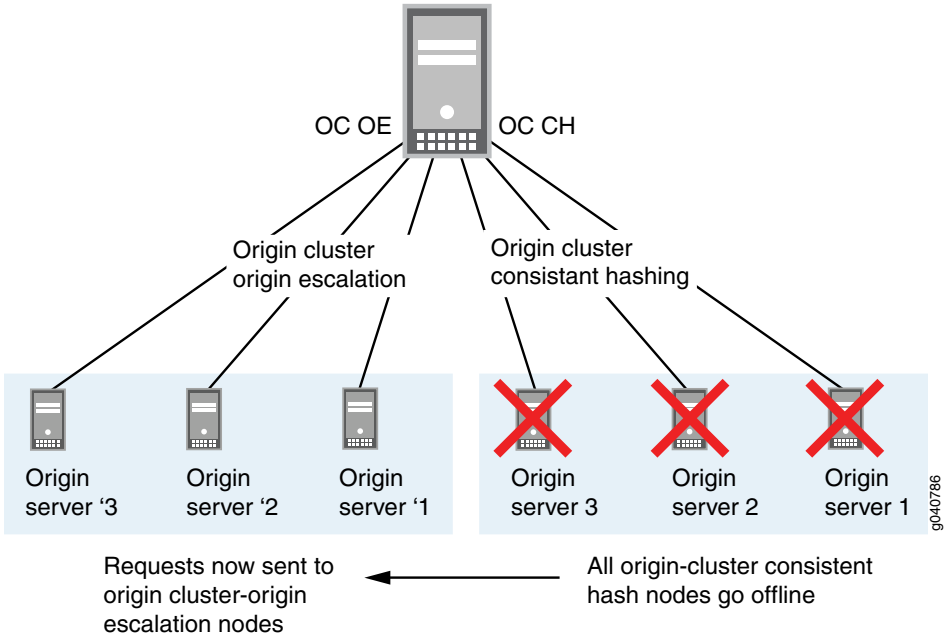
Cluster Stacking

Within a Media Flow Controller, you can define up to three origin type clusters in a single namespace. A typical configuration uses origin cluster-cluster hash (OC-CH) to distribute objects among the nodes. However, if the cluster hash nodes all go offline for any reason, origin-cluster-origin escalation (OC-OE) can be configured to satisfy requests. This type of configuration is called cluster stacking.

The advantage of stacking OC-CH with OC-OE is that, if all active nodes using the OC-CH go offline, the Media Flow Controller determines that the nodes are offline (through its heartbeat status network) and uses OC-OE to satisfy requests instead of returning an error.

In the origin server example shown in Figure 13 on page 218, two types of origin servers are defined: origin cluster-consistent hash (Origin servers 1 through 3) and origin cluster-origin escalation (Origin server prime 1 through 3). Optionally, a second OC-OE (not shown in the figure) can be configured so that there is an alternate OC-OE in case all nodes of the first OC-OE go offline. If all nodes of the OC-CH go offline (as shown in the figure), then the Media Flow Controller uses OC-OE to escalate to the origin escalation server. If a second OC-OE is configured, and all nodes of the OC-CH plus all nodes of the first OC-OE go offline, the Media Flow Controller executes the second OC-OE when other origin servers go offline. When alternate origin clusters are defined and executed to satisfy requests, the method is called origin cluster stacking.

Figure 13: Example of Cluster Stacking



Instead of defining a second origin cluster-origin escalation in the stack, you can also define a weighted round-robin origin cluster. You can define and stack any combination of OC-CH, OC-OE, or weighted round-robin in a single Media Flow Controller namespace.

Related Documentation

- Clustering Overview on page 209
- Consistent Hash for Mapping Requests to Servers on page 210
- Origin Failover and Escalation on page 212
- Cluster Weighted Round-Robin Load Distribution on page 213
- Configuring Origin Cluster Weighted Round-Robin on page 216
- Transparent Cluster Overview on page 218
- Configuring Transparent Clustering on page 220
- Media Flow Controller Server Map Overview on page 223

Transparent Cluster Overview

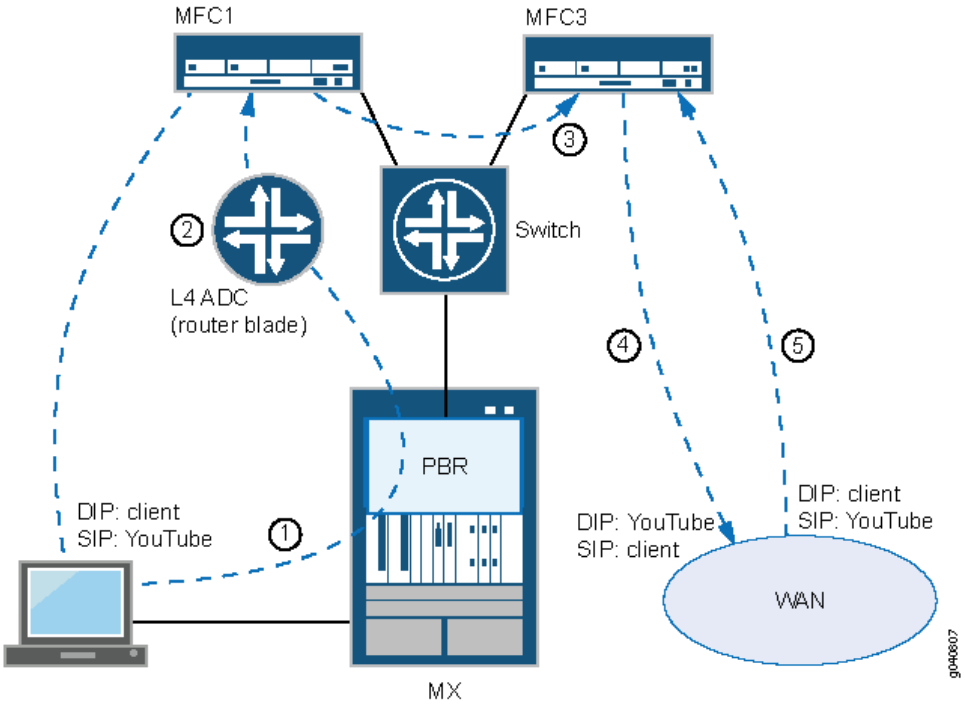
Transparent cluster extends Origin clustering to utilize transparent proxy. It reduces the amount of network hardware required, as this strategy eliminates the need for a Layer 7 load balancer. As long as the router supports port range traffic forwarding, response traffic from origin servers can be sent directly to the Media Flow Controllers, bypassing the load balancers and reducing cost and latency.

This topic covers:

- Double-Tier Cluster on page 219

Double-Tier Cluster

Figure 14: Double-Tier Cluster Transparent Proxy Using a Server Load Balancer with an MX Series Router



Using a two-tiered topology, the router, Layer 4 load balancer, and Tier 1 and Tier 2 Media Flow Controllers are all connected through a switch, so traffic can flow directly from one box to another, depending on the routing configuration on each box.

1. On the router, request traffic is forwarded to one of the load balancers, using filter-based forwarding, or policy-based routing. Response traffic from the origin servers bypasses the load balancer and goes directly to a Media Flow Controller using a port range filter.
2. Layer 4 load balancers receive request traffic from the router and forward requests to one of the first tier Media Flow Controllers. The connections are persistent: if one Media Flow Controller goes down TCP connections to other Media Flow Controllers are not affected.
3. If the request matches a namespace not configured for clustering, it is processed normally. If the request matches a namespace with clustering configuration set up:
   a. If a decision is made to tunnel the request, it is sent directly to the origin server.
   b. For non-tunneling traffic, the Media Flow Controller checks its own memory cache (no disk cache checking). For a cache hit, it returns the response to the client.

c. For memory cache miss, a consistent hash algorithm is used to calculate the target node. The request is then forwarded to the target node Media Flow Controller (Tier 2 Media Flow Controller).

4. The target node Media Flow Controller processes the request as a normal Media Flow Controller does without clustering.

Related Documentation

- Clustering Overview on page 209
- Consistent Hash for Mapping Requests to Servers on page 210
- Origin Failover and Escalation on page 212
- Cluster Weighted Round-Robin Load Distribution on page 213
- Configuring Origin Cluster Weighted Round-Robin on page 216
- Cluster Stacking on page 217
- Configuring Transparent Clustering on page 220
- Media Flow Controller Server Map Overview on page 223

Configuring Transparent Clustering

The steps below provide an overview of the process used to configure a transparent proxy cluster to handle requests and responses appropriately for a two-tiered topology. This configuration involves a cluster of Media Flow Controllers, load balancers, as well as Juniper Networks routers. This topic describes only Media Flow configuration.

The example instructions below create a cluster config file, l7cluster1.xml, which is used to define server map l7cluster, which in turn defines cluster cl-l7. Finally, cluster cl-l7 is added to the namespace responsible for the tier 1 cluster.

To configure transparent proxy clustering:

1. Create a server map.

   The server map is the single source that identifies all of the nodes of the proxy cluster. The server map is used by the first-tier Media Flow Controllers only. In a single-tier topology, the server map lists all Media Flow Controllers, as all Media Flow Controllers are treated equally. Each can serve the content from its disk cache or can proxy the request to another Media Flow Controller or to the Origin server. In a two-tiered topology, the server map lists only the second- tier Media Flow Controllers. This file can be located on any HTTP server accessible to the Media Flow Controller. For a more detailed explanation of server maps and their uses, see "Media Flow Controller Server Map Overview" on page 223.

   File l7cluster1.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE ClusterMap SYSTEM "ClusterMap.dtd">
<ClusterMap>
  <Header>
    <Version>1.0</Version>
    <Application>MapXML</Application>
```



```
</Header>
<ClusterMapEntry>
  <Node>NodeName1</Node>
  <IP>10.157.43.138</IP>
  <Port>80</Port>
  <Options>heartbeatpath=:8080/cmm-node-status.html</Options>
</ClusterMapEntry>
<ClusterMapEntry>
  <Node>NodeName2</Node>
  <IP>10.157.34.104</IP>
  <Port>80</Port>
  <Options>heartbeatpath=:8080/cmm-node-status.html</Options>
</ClusterMapEntry>
</ClusterMap>
```

Use the file to specify the each cluster node and its attributes (See Table 21 on page 221.)

Table 21: Cluster Map Tag Descriptions

Cluster Map Tag	Description
Node	Name of the node.
IP	Node's IP address.
Port	The node's port.
Options	Sets options for the node.

2. Enable transparent proxy and set port ranges.
   a. On the Media Flow Controller, enable transparent proxy.

      delivery protocol http transparent *interface_name_* enable

   b. Configure the port range.

      (config) # **network connection transparent-proxy *port-range***

   c. Configure a server map from the file created in Step 1.

      (config)# **server-map l7cluster**

      (config)# **format-type cluster-map**

      (config)# **file-url http://10.157.42.52/testresults/yunleiy/l7cluster1.xml refresh-interval 300**

      (config)# **node-monitoring heartbeat interval 100000**

   d. Create a cluster and bind it to the newly created server map.

      (config)# **cluster create cl_l7 server-map l7cluster**

   e. Create a namespace and add the cluster to it.

      (config)# **namespace *tier1***

(config)# **match uri /**

(config)# **domain any**

(config)# **origin-server http follow dest-ip use-client-ip**

(config)# **delivery protocol http origin-request include-header request-ip request-port**

(config)# **status active**

f. Add cluster cl_l7.

(config)# **cluster cl_l7 request-routing method consistent-hash-proxy**

(config)# **cluster cl_l7 request-routing method consistent-hash-proxy cluster-hash complete-url**

(config)# **cluster cl_l7 request-routing method consistent-hash-proxy proxy-addr-port 255.255.255.255:65535**

(config)# **cluster cl_l7 request-routing method consistent-hash-proxy proxy-local allow**

(config)# **cluster cl_l7 request-routing load-balance load-threshold 80**

(config)# **cluster cl_l7 request-routing load-balance replicate-mode least-loaded**

3. On a Tier 2 MFC, use the normal transparent proxy configuration for the same namespace, but also configure the port range.

(config)# **delivery protocol http transparent <*if-name*> enable**

(config)# **network connection transparent-proxy port-range <*min-range*> <*max-range*>**

Related Documentation

- Clustering Overview on page 209
- Consistent Hash for Mapping Requests to Servers on page 210
- Origin Failover and Escalation on page 212
- Cluster Weighted Round-Robin Load Distribution on page 213
- Configuring Origin Cluster Weighted Round-Robin on page 216
- Cluster Stacking on page 217
- Transparent Cluster Overview on page 218
- Media Flow Controller Server Map Overview on page 223

CHAPTER 9

Configuring Media Flow Controller Server Maps

Media Flow Controller Server Map Overview

Media Flow Controller can use an XML file to specify various origin fetch schemes. For example. associating a server map with a namespace will ensure that all content meeting the criteria of a specific namespace will be routed according to the policies defined in the server map. The server map file can reside in an external server. The format types and syntax for the XML files to which the **server-map** command provides access are described in this chapter. When you have defined the XML file, from the Media Flow Controller CLI for **namespace**, configure the **origin-server** ***protocol*** **server-map** option with the **format-type** and file URL of the XML mapping file that you defined.

The following limitations apply to the use of server maps with namespaces:

- There can be no more than 256 server maps.
- No more than 64 namespaces can be associated with a server map.

Server Map Format Types

Media Flow Controller provides four **server-map format-type** options:

- **host-origin-map**—Use this server map type to configure multiple HTTP origin servers. The hostname (Host header) in the incoming request is matched to the Host entry you define in the XML file. The matched XML **Host** entry denotes the target origin server hostname and port using the associated XML **Origin** and **Port** entries. You cannot use another **server-map** with this **format-type**.
- **cluster-map**—Use this server map type to distribute incoming requests across a cluster of origin servers using consistent hashing to bind objects to nodes, or to create a cluster Layer 7 redirect configuration. You can use this **format-type** with the **origin-escalation-map** to create a server map hierarchy, or you can use this map alone.
- **origin-escalation-map**—Use this server map type to configure multiple redundant HTTP origin servers for failover protection. Requests are sequentially initiated to specific origin servers based on a configured weight, until the request is satisfied or all available origin servers have been tried. You can use this **format-type** with **cluster-map** to create a server map hierarchy, or you can use this map alone.
- **nfs-map**—Use this server map type to configure multiple NFS publishing points for the origin. You cannot use another **server-map** with this **format-type**.

NOTE: For cluster-map and origin-escalation-map only, you can assign up to three formats, in any combination (together or separately), as needed.

Related Documentation

- Configuring Server Maps (CLI) on page 232
- Origin Server Load Distribution and Failover on page 225
- *logical-interface (DDoS Flow Detection)*
- Load Feedback API on page 235

Creating the cluster-map XML File

Use the **server-map format-type cluster-map** to distribute incoming requests across a cluster of origin servers. This server-map type calls for specific definition of a consistent hashing scheme used to bind objects to nodes. For background information on consistent hashing, see http://www8.org/w8-papers/2a-webserver/caching/paper2.html.

See **server-map** in the *Media Flow Controller CLI Command Reference* for CLI details.

Consistent hash cluster requirements include:

- A cluster is one Level 1 node and greater than or equal to one Level 2 nodes.
- The Level 1 node must be a Media Flow Controller operating as a reverse proxy.

- Sample cluster-map XML File for Origin-Based Clusters on page 225
- Server Map Example: cluster-map DTD on page 225

Sample cluster-map XML File for Origin-Based Clusters

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE ClusterMap SYSTEM "ClusterMap.dtd">
<ClusterMap>
  <Header>
   <Version>1.0</Version>
   <Application>MapXML</Application>
  </Header>
  <ClusterMapEntry>
   <Node>NodeName1</Node>
   <IP>172.19.172.52</IP>
   <Port>80</Port>
   <Options>heartbeatpath=/root</Options>
  </ClusterMapEntry>
  <ClusterMapEntry>
   <Node>NodeName2</Node>
   <IP>172.19.172.53</IP>
   <Port>81</Port>
   <Options>heartbeatpath=/root</Options>
  </ClusterMapEntry>
</ClusterMap>
```

Server Map Example: cluster-map DTD

In the following Document Type Definition (DTD) for **cluster-map**, "PCDATA" indicates data that the XML parser fills in after reading your file. You can use this to validate your XML file as described. For an example host-origin-map.xml file, see "Creating the cluster-map XML File" on page 224.

```
<!-- ClusterMap 1.0 DTD, Copyright (c) 2010 by Juniper Networks, Inc -->
<!ELEMENT ClusterMap (Header*, ClusterMapEntry*)>
  <!ELEMENT Header (Version, Application)>
   <!ELEMENT Version (#PCDATA)>
   <!ELEMENT Application (#PCDATA)>
  <!ELEMENT ClusterMapEntry (Node, IP, Port, Options?)>
   <!ELEMENT Node (#PCDATA)>
   <!ELEMENT IP (#PCDATA)>
   <!ELEMENT Port (#PCDATA)>
   <!ELEMENT Options (#PCDATA)>
```

Related Documentation

- Configuring Server Maps (CLI) on page 232
- *logical-interface (DDoS Flow Detection)*
- Media Flow Controller Server Map Overview on page 223

Origin Server Load Distribution and Failover

Media Flow Controller lets you to create an origin-server node map for origin escalation.

If the target origin server fails or returns a configured HTTP code requiring escalation, another configured origin server is automatically chosen. This is done with the creation of a server map (**format-type origin-escalation-map**) that is then associated with a **namespace**.

In the cache-miss case, the **origin-escalation-map** server map is consulted.

Origin escalation is a configuration consisting of <N> origin servers which are logically viewed as one, where requests are sequentially initiated to specific origin servers (based on a configured **weight**), until the request is satisfied or all known available origin servers at request initiation time have been tried. An origin-server request is re-initiated to the next configured origin server (escalation) when network connectivity errors are received or when a specific, configured, origin server response code is received (for example, HTTP 404). The origin servers are tried in the order in which they are listed in the XML origin escalation map file, and the order in which the maps are added to the **namespace** denotes the order in which they are read. You can add up to three origin escalation maps to a namespace.

Additionally, you can create a simple **host-origin-map** server map for load distribution, but not for failover.

Related Documentation

- Media Flow Controller Server Map Overview on page 223
- Creating the origin-escalation-map XML File on page 226
- Configuring Server Maps (CLI) on page 232

Creating the origin-escalation-map XML File

The **origin-escalation-map** functionality determines which defined origins are online. If an origin is not online, the process automatically moves to the next defined origin, based on a defined **weight**. Under origin escalation, the specified **weight** of each origin defined in the map denotes the order in which requests are tried until a success or all nodes have been tried. Escalation occurs when a network connectivity error is encountered, or when an origin returns an HTTP return code specified in the XML file configuration.

See **server-map** in the *Media Flow Controller CLI Command Reference* for CLI details.

Origin escalation requirements include:

- All origin servers are viewed as a single entity where any origin server can resolve a miss or handle a validate request.
- All origin servers are monitored using a periodic heartbeat (HTTP request) insuring that escalation only occurs to members that are currently online.
- Origin server HTTP response codes that result in escalation can be configured on a per origin server basis.
- Origin escalation applies only to a Media Flow Controller reverse proxy configuration.

- Example: origin-escalation map XML file on page 227
- Create an origin-escalation-map XML file on page 227
- Server Map Example: origin-escalation-map DTD on page 228

Example: origin-escalation map XML file

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE OriginEscalationMap SYSTEM "OriginEscalationMap.dtd">
<OriginEscalationMap>
  <Header>
   <Version>1.0</Version>
   <Application>MapXML</Application>
  </Header>
  <OriginEscalationMapEntry>
   <Origin>dest.xxx.com</Origin>
   <Port>80</Port>
   <Options>heartbeatpath=/hello.html,weight=1,
    http_response_failure_codes=404;500</Options>
  </OriginEscalationMapEntry>
  <OriginEscalationMapEntry>
   <Origin>dest.yyy.com</Origin>
   <Port>80</Port>
   <Options>heartbeatpath=/hello.html,weight=2,
    http_response_failure_codes=404;500</Options>
  </OriginEscalationMapEntry>
  <OriginEscalationMapEntry>
   <Origin>dest.zzz.com</Origin>
   <Port>80</Port>
   <Options>heartbeatpath=/hello.html,weight=3,
    http_response_failure_codes=404;500;503</Options>
  </OriginEscalationMapEntry>
</OriginEscalationMap>
```

Create an origin-escalation-map XML file

To create an **origin-escalation-map** XML file:

1. Open a text editor and copy and paste the above example into the file. You can define up to 64 nodes in a single **origin-escalation-map**. If you want to use more than 64 nodes for origin escalation, you can define additional maps (up to 3 server maps can be added to a namespace). Change all the tag variables and add entries as needed. You can use the following tags:
   - **OriginEscalationMap**—Parent tag for **Header** and **OriginEscalationMapEntry** tags.
     - **Header**—Parent tag for **Version** and **Application** tags.
       - **Version**—MapXML utility version: **1.0**.
       - **Application**—Type of Media Flow Controller application: **MapXML**.
     - **OriginEscalationMapEntry**—Parent tag for **Origin**, **Port**, and **Options** tags.
       - **Origin**—Domain name or IP address of the origin server.
       - **Port**—TCP port of the origin server.
       - **Options**:
         - **heartbeatpath**—Relative URI to use to heartbeat the node.
         - **http_response_failure_codes** —Codes that trigger escalation. The **http_response_failure_codes** are checked when an incoming request is

processed and response from origin is received. The **http_response_failure_codes** do not affect origin escalation. For example, 404 or 503 responses are considered healthy from the node monitoring perspective.

- **weight=<integer>**—Order of escalation; highest priority is the lowest value.

Do not change the **Version** from **1.0** or the **Application** from **MapXML** without explicit, new instructions. These values must be present or the XML file will be rejected.

2. Validate your XML against the DTD by running the following under Linux:

   xmllint --noout --dtdvalid *DTD filename XML filename*

   Example:

   xmllint --noout --dtdvalid ./dtd/OriginEscalationMap.dtd OriginEscalationMap.xml

   Ignore the following xmllint warning:

   OriginEscalationMap.xml:2: warning: failed to load external entity
   "OriginEscalationMap.dtd"

Server Map Example: origin-escalation-map DTD

In the following Document Type Definition (DTD) for **origin-escalation-map**, " PCDATA" indicates data that the XML parser fills in after reading your file. You can use this to validate your XML as described. For an example *host-origin-map.xml* file, see "Creating the origin-escalation-map XML File" on page 226.

```
<!-- OriginEscalationMap 1.0 DTD, Copyright (c) 2010 by Juniper Networks, Inc
-->
<!ELEMENT OriginEscalationMap (Header*, OriginEscalationMapEntry*)>
  <!ELEMENT Header (Version, Application)>
   <!ELEMENT Version (#PCDATA)>
   <!ELEMENT Application (#PCDATA)>
  <!ELEMENT OriginEscalationMapEntry (Origin, Port, Options?)>
   <!ELEMENT Origin (#PCDATA)>
   <!ELEMENT Port (#PCDATA)>
   <!ELEMENT Options (#PCDATA)>
```

Related Documentation

- Configuring Server Maps (CLI) on page 232
- Origin Server Load Distribution and Failover on page 225
- Media Flow Controller Server Map Overview on page 223

Creating the host-origin-map XML File

Use the **server-map format-type host-origin-map** XML file to map the incoming (target origin) Host header value to a specified origin server and (optionally) port. No check is

made to determine if an origin is online when requested. There is no maximum on how many origins you configure in this file.

- Example: host-origin-map XML file on page 229
- Create the host-origin-map XML file on page 229
- Server Map Example: host-origin-map DTD on page 230

Example: host-origin-map XML file

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE HostOriginMap SYSTEM "HostOriginMap.dtd">
<HostOriginMap>
  <Header>
   <Version>1.0</Version>
   <Application>MapXML</Application>
  </Header>
  <HostOriginEntry>
   <Host>host1.xxx.yyy.com</Host>
   <Origin>host1.com</Origin>
   <Port>80</Port>
  </HostOriginEntry>
  <HostOriginEntry>
   <Host>host2.aaa.bbb.com</Host>
   <Origin>host2.com</Origin>
   <Port>80</Port>
  </HostOriginEntry>
  <HostOriginEntry>
   <Host>host3.ccc.ddd.com</Host>
   <Origin>host3.com</Origin>
   <Port>80</Port>
  </HostOriginEntry>
 </HostOriginMap>
```

Create the host-origin-map XML file

To create the **host-origin-map** XML file:

1. Open a text editor and copy and paste the above example into the file. Change all the tag variables and add entries as needed. You can use the following:
   - **HostOriginMap**—Parent tag for the **Header** and **HostOriginEntry** tags.
     - **Header**—Parent tag for the **Version** and **Application** tags:
       - **Version**—MapXML utility version: **1.0**.
       - **Application**—Type of Media Flow Controller application: **MapXML**.
     - **HostOriginEntry**—Parent tag for **Host**, **Origin**, **Port**, and **Options** tags.
       - **Host**—The incoming request HOST header (target origin).
       - **Origin**—The origin server to use to resolve cache misses.
       - **Port**—The origin server port to use to resolve cache misses.
       - **Options**—Not supported.

Do not change the **Version** from **1.0** or the **Application** from **MapXML** without explicit, new instructions. These values must be present or the XML file will be rejected.

Do not change the **Version** from **1.0** or the **Application** from **MapXML** without explicit, new instructions. These values must be present or the XML file will be rejected.

2. Validate your XML syntax against the DTD by running the following under Linux:

   xmllint --noout --dtdvalid *DTD filename XML filename*

   Example:

   xmllint --noout --dtdvalid ./dtd/HostOriginMap.dtd HostOriginMap.xml

   Ignore the following xmllint warning:

   HostOriginMap.xml:2: warning: failed to load external entity "HostOriginMap.dtd"

Server Map Example: host-origin-map DTD

In the following Document Type Definition (DTD) for **host-origin-map**, " PCDATA" indicates data that the XML parser fills in after reading your file. You can use this to validate your XML as described.

```
<!-- HostOriginMap 1.0 DTD, Copyright (c) 2010 by Juniper Networks, Inc. -->
<!ELEMENT HostOriginMap (Header*, HostOriginEntry*)>
  <!ELEMENT Header (Version, Application)>
   <!ELEMENT Version (#PCDATA)>
   <!ELEMENT Application (#PCDATA)>
  <!ELEMENT HostOriginEntry (Host, Origin, Port)>
   <!ELEMENT Host (#PCDATA)>
   <!ELEMENT Origin (#PCDATA)>
   <!ELEMENT Port (#PCDATA)>
   <!ELEMENT Options (#PCDATA)>
```

Related Documentation

- Configuring Server Maps (CLI) on page 232
- Origin Server Load Distribution and Failover on page 225
- Media Flow Controller Server Map Overview on page 223

Creating the nfs-map XML File

Use the **server-map format-type nfs-map** XML file to provide a list of NFS mount points mainly, hostname and publishing point, and some other information.

Example **nfs-map** XML file:

```
<response scode="0" scode_description="success!">
<PUBLISHINGPOINTS INTERVAL-SEC="3600">
<MISSINGFILE PATH="C:\path1\path2\path3\FileNotFound.wma"/>
<INVALIDPUBLISHINGPOINT PATH="C:\path1\path2\path3\StorageOffline.wma"/>
<PUBLISHINGPOINT NAME="name1" PATH="\\<name>.<name>.<name>.com\name1"/>
<PUBLISHINGPOINT NAME="name2" PATH="\\<name>.<name>.<name>.com\name2"/>
<PUBLISHINGPOINT NAME="name3" PATH="\\<name>.<name>.<name>.com\name3"/>
<PUBLISHINGPOINT NAME="name4" PATH="\\<name>.<name>.<name>.com\name4"/>
</PUBLISHINGPOINTS>
</response>
```

NOTE: The Media Flow Controller NFS origin server map uses a proprietary parser to extract data from the XML file, which file must use the tags and format described. There is no DTD available to use to validate this type of map.

To create the **nfs-map** XML file:

1. Open a text editor and copy and paste the above example into the file. Change all the tag variables and add entries as needed. You can use the following tags:
   - **Response Code**—The standard response code when indicating success in calling the NFS service. You can modify the description.
   - **PublishingPoints Interval-Sec**—The polling interval. You can modify the number of seconds. This can also be set in the Media Flow Controller CLI with the **server-map** command. The CLI setting overrides the XML file setting.
   - **MissingFile Path**—A file to use in the case of a missing file path. This is needed only if you will be handling streaming of Windows Media content.
   - **InvalidPublishingPoint Path**—A file to use in the case of an invalid publishing point. This is needed only if you will be handling streaming of Windows Media content.
   - **PublishingPoint Name**—The NFS mount point for that file path. Example:

     <PUBLISHINGPOINT NAME="*name*"
     PATH="*name.name.name*.com*mount_point*" />

     CAUTION: The defined publishing point names are expected to be the first component in the incoming URI request. For example, if you define a publishing point name “ p12s34” (<PUBLISHINGPOINT NAME=” p12s34” >), then incoming URI requests must include that name in the first part of the URI for the mapping of that publishing point to be successful; for example, **http:// p12s34 .example.com/path/filename.**

Related Documentation

- Creating the host-origin-map XML File on page 228
- Creating the origin-escalation-map XML File on page 226
- Configuring Server Maps (CLI) on page 232
- Media Flow Controller Server Map Overview on page 223

Configuring Server Maps (CLI)

In the Media Flow Controller CLI, after you have created the necessary XML file, you define the server map with the **server-map** command, and then assign it to a defined **namespace**.

Configuring the Server Map (CLI)

To define a server map:

1. Create the server map with a name.

   server-map *name*

   You enter prefix mode.

2. Set the **format-type**:

   - If this server map will be assigned to a namespace with **origin-server nfs**, and you want multiple NFS origin servers, set **format-type** to **nfs-map**.
   - If this server map will be assigned to a namespace with **origin-server http**, and you want multiple HTTP origin servers (without consistent hashing), set the **format-type** to **host-origin-map**.
   - If this server map will be assigned to a namespace with **origin-server http**, and you want to distribute requests across a cluster of origin servers using consistent hashing to bind objects to nodes, set the **format-type** to **cluster-map**.

     The prefix prompt is re-displayed.

   - If this server map will be assigned to a namespace with **origin-server http**, and you want to configure origin escalation, set the **format-type** to **origin-escalation-map**.

     format-type *type*

   The prefix prompt is re-displayed.

3. If you are using a **cluster-map** or **origin-escalation-map**, optionally set additional **allowed-fails** and **node-monitoring** options:

   - **allowed-fails**—Specify how many request failures are allowed before the node is declared down. The default is **3**, the minimum allowed value is **0** (zero), the maximum value is **32**.
   - **node-monitoring**—Set node monitoring options:
     - **connect-timeout**—Specify the allowable time, in milliseconds, for the socket connect to complete.
     - **heartbeat-interval**—Specify the time, in milliseconds, for nodes to wait before sending a " heartbeat" signal to the other nodes indicating availability status.

- **read-timeout**—Specify the allowable time, in milliseconds, for the socket read to complete after the connection is established.

  allowed-fails *integer* node-monitoring *option value*...

The prefix prompt is re-displayed.

4. Set a **file-url** for the server-map. This tells Media Flow Controller from where to fetch the XML mapping file. Also, set how often Media Flow Controller refreshes the XML file. The valid refresh-interval value is 0, that means no refresh, or the value must be between the range 300 seconds and 86400 seconds. The default **refresh-interval** is **300** seconds. The refresh-interval can also be set in the XML mapping file; the CLI setting overrides the XML file setting.

   file-url <URL> refresh-interval <time>

   Media Flow Controller immediately issues a request to the file-url you configured to verify the XML file. If there is a problem; for example, Media Flow Controller cannot find the file, an error message appears. Otherwise, a confirmation of the file appears.

5. Assign the **server-map** to a **namespace**, and set options as applicable. You can assign up to three server maps of the **origin-escalation** or **cluster-map** type and you can use the same server map in multiple namespaces.

   namespace *name* match uri *uri-prefix* origin-server *protocol* server-map *name*

Example: cluster-map Configuration

```
MFC (config) # server-map clusterSF
MFC (config server-map clusterSF) # format-type cluster-map
MFC (config server-map clusterSF) # file-url http://example.com/nfs/path/
cluster.xml refresh-interval 300
MFC (config server-map clusterSF) # exit
MFC (config) # show server-map clusterSF
  Server-map : clusterSF
  Format-type : cluster-map
  Map File : http://example.com/nfs/path/cluster.xml
  Refresh Interval : 300
MFC (config) # namespace newCM
MFC (config namespace newCM) # match uri /
MFC (config namespace newCM) # origin-server http server-map clusterSF
MFC (config namespace newCM) # exit
MFC (config) #
```

Example: origin-escalation-map Configuration

```
MFC (config) # server-map OE1
MFC (config server-map OE1) # format-type origin-escalation-map
MFC (config server-map OE1) # file-url http://example.com/nfs/path/
originEscalation.xml refresh-interval 300
MFC (config server-map OE1) # exit
MFC (config) # show server-map OE1
  Server-map : OE1
  Format-type : origin-escalation-map
  Map File : http://example.com/nfs/path/originEscalation.xml
  Refresh Interval : 60
MFC (config) # namespace newOE
MFC (config namespace newOE) # match uri /
MFC (config namespace newOE) # origin-server http server-map OE1
MFC (config namespace newOE) # exit
MFC (config) #
```

Example: HTTP host-origin-map Configuration

```
MFC (config) # server-map newMap
MFC (config server-map newMap) # format-type host-origin-map
MFC (config server-map newMap) # file-url http://example.com/vod/
newHostOriginMap.xml refresh-interval 9000
MFC (config server-map newMap) # exit
MFC (config) # show server-map newMap
  Server-map : newMap
  Format-type : host-origin-map
  Map File : http://example.com/vod/newHostOriginMap.xml
  Refresh Interval : 9000
MFC (config) # namespace newTest
MFC (config namespace newTest) # match uri /
MFC (config namespace newTest) # origin-server http server-map newMap
MFC (config namespace newTest) # exit
MFC (config) #
```

Related Documentation

- Creating the host-origin-map XML File on page 228
- Creating the origin-escalation-map XML File on page 226
- Creating the nfs-map XML File on page 230

- Media Flow Controller Server Map Overview on page 223

Load Feedback API

Load Feedback Application Programming Interface (API), allows your load balancing equipment to receive and monitor load information from Media Flow Controllers and make appropriate decisions about forwarding traffic to those Media Flow Controllers.

The API consists of an HTTP interface that enables registered applications external to Media Flow Controller to interact directly with Media Flow Controllers to request and receive system load information. The load information is presented using XML format. The load feedback API runs on Media Flow Controllers and can be integrated with your load monitoring systems, such as content routers and Global Server Load Balancers (GSLBs), which use this load information to decide whether or not to forward additional traffic to a particular Media Flow Controller. The GSLB can then determine to which Media Flow Controller to forward the request based on the load information it receives.

The Media Flow Controller XML Schema has load counter information pertaining to:

- Global device or server resource usage level counters, such as CPU utilization, concurrent TCP session, and throughput.
- Resource pool utilization counters of HTTP applications, such as throughput and concurrent TCP session counters

The load feedback API uses the HTTP GET method to exchange XML load content. The HTTP GET method is used for probe requests. Probe responses include the HTTP response code, response message, content length, content type, and load counter information, contained in a single XML formatted message.

NOTE: **Because the polling interval can have a performance impact, we recommend 5 seconds as the polling interval.**

The messaging protocol allows the external application to control certain parameters. It can filter and select particular sections of the load information to view, request a particular set of load counter information, and select a moving average view or a snapshot view of the information. This can be achieved by customizing the XML schema. The load feedback module only listens on the base IP address of an interface. If an interface has multiple IP addresses, ensure the IP address used by any other software exercising the load feedback API is the base IP address of the interface and not an alias address.

The following controls available in Media Flow Controller are:

- Window length
- Sampling duration
- Maximum TCP connection rate
- Maximum disk I/O operations

Successful probe responses from Media Flow Controller include:

- "200 OK" response status
- Content-type header as *application/XML*
- Content-length header containing the length of the response
- Body of the message represented as an XML document

Unsuccessful probe responses from Media Flow Controller include:

- 4xx or 5xx response status and message
- Content-type header as *text/HTML*
- Content-length header containing the length of the response
- Empty message body with a 404 error code returned in case of error

For a summary of Media Flow Controller load counters, see Table 22 on page 236. The Media Flow Controller does not calculate an aggregate load metric; the load metric is calculated by the registered external application based on the values of the individual load counters it receives.

Table 22: Load Counters

Resource	Description	Global-Level Counters	VM-Level Counters	Resource Pool
Bandwidth	Delivery throughput (bytes per second).	X	X	X
Concurrent Sessions	Number of active TCP user sessions.	X	–	X
Transactions per Second	Number of HTTP transactions per second.	X	–	–
TCP Connections per Second	Number of TCP user connections per second.	X	–	–
CPU	CPU utilization of all the cores in the system and CPU utilization of the core engine (maximum of CPU utilization of the core engine threads is reported).	X	X	–

Table 22: Load Counters *(continued)*

Resource	Description	Global-Level Counters	VM-Level Counters	Resource Pool
Disk Usage	How busy the caching disks are from an application standpoint. One counter exists for each cache tier (SATA, SAS, and SSD). Disk usage is represented as a percentage (from 0 to 100), where 0 is not busy to 100 which indicates maximum capacity. When disk usage reaches 100 percent, Session Admission Control (SAC) starts.	X	–	–
Disk Utilization	Refers to all the disks in a Media Flow Controller server. One counter exists for each caching and non-caching disk. Disks are labeled using standard Linux disk labels (sda, sdb, and so forth). Disk usage is represented as a percentage (from 0 to 100), where 0 percent is not busy to 100 percent which indicates maximum capacity. One MaxDiskUtil counter exists with the maximum threshold set to 100.	X	–	–
RAM Utilization	The percentage of RAM or attribute cache buffers currently used for transit. When this counter reaches 100 percent, a SAC error occurs.	X	–	–

Media Flow Controller pre-reads objects in the cache when a Media Flow Controller stops and restarts again; for example, when a caching and delivery subsystem restarts or system reboot occurs. During pre-read, the Media Flow Controller generates an index of the cached objects. The pre-read state indicates the status of the pre-read operation with the following values:

- -1 indicates that the disk tier is not present in the server.
- 0 indicates that pre-read is in progress. (The system is up, but not ready to serve objects from the cache. Requests received from users are tunneled to the origin server.)
- 1 indicates that the pre-read is complete. (The system is ready to serve objects from the cache.)

Related Documentation

- Load Feedback API XML Schema Configuration on page 238
- Example Load Feedback XML Schema Output on page 242

Load Feedback API XML Schema Configuration

Load Feedback Application Programming Interface (API), allows your load balancing equipment to receive and monitor load information from Media Flow Controllers and make appropriate decisions about forwarding traffic to those Media Flow Controllers. It is a single point of data collection and statistics computation and provides consistent data reporting across all external interfaces.

The API consists of an HTTP interface that enables registered applications external to Media Flow Controller to interact directly with Media Flow Controllers to request and receive system load information. The load information is presented using XML format. The load feedback API runs on Media Flow Controllers and can be integrated with your load monitoring systems, such as content routers and Global Server Load Balancers (GSLBs), which use this load information to decide whether or not to forward additional traffic to a particular Media Flow Controller. The GSLB can then determine to which Media Flow Controller to forward the request based on the load information it receives.

Interaction of the API with external applications relies primarily on the HTTP GET method to probe Media Flow Controllers for load feedback and return relevant load-level information to external applications requesting it.

Load information is presented in XML format as a response to the HTTP GET requests.

Media Flow Controller response includes server-level, virtual machine-level, and native application-level counters, delivered in a single XML Schema file response to the registered external application.

The external application, which is integrated with content routers, load balancers, or other load monitoring systems, uses this information to determine whether or not to forward additional traffic to the Media Flow Controller.

The following is an example of an XML Schema to be included in the response to the load-level feedback probes from the global server load balancer, content router, or other load monitoring device. You can copy this XML Schema example, modify it to meet your needs, and deploy it as a starting point.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema version="1.0"
    targetNamespace="http://www.juniper.net/mfc/schema/lf"
    xmlns="http://www.juniper.net/mfc/schema/lf"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">
  <xs:complexType name="cpu-usage-type">
    <xs:attribute name="id" type="xs:nonNegativeInteger" use="required"/>
    <xs:attribute name="util" type="xs:float" use="required"/>
  </xs:complexType>
  <xs:complexType name="disk-usage-type">
    <xs:attribute name="id" type="xs:string" use="required"/>
```

```
    <xs:attribute name="util" type="xs:float" use="required"/>
  </xs:complexType>
  <xs:complexType name="cpu-info">
    <xs:sequence>
      <xs:element name="cpu" type="cpu-usage-type" minOccurs="1"
    maxOccurs="unbounded">
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="disk-info">
    <xs:sequence>
      <xs:element name="disk" type="disk-usage-type" minOccurs="1"
    maxOccurs="unbounded">
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="system-type">
    <xs:sequence>
      <xs:element minOccurs="0" name="hostName" type="xs:string"/>
      <xs:element name="cpuInfo" type="cpu-info" minOccurs="1"
    maxOccurs="unbounded"/>
      <xs:element name="maxCPUUsage" type="xs:nonNegativeInteger"/>
      <xs:element name="maxDiskUsage" type="xs:nonNegativeInteger"/>
      <xs:element name="ifBandwidth" type="xs:nonNegativeInteger"/>
      <xs:element name="maxIfBandwidth" type="xs:nonNegativeInteger"/>
      <xs:element name="diskInfo" type="disk-info" minOccurs="1"/>
      <xs:element name="httpTPS" type="xs:nonNegativeInteger"/>
      <xs:element name="maxHttpTPS" type="xs:nonNegativeInteger"/>
      <xs:element name="tcpConnRate" type="xs:integer"/>
      <xs:element name="maxTcpConnRate" type="xs:nonNegativeInteger"/>
      <xs:element name="tcpActiveConn" type="xs:nonNegativeInteger"/>
      <xs:element name="maxTcpActiveConn" type="xs:nonNegativeInteger"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="vma-type">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="1" name="vmaName"
type="xs:string"/>
     <xs:element maxOccurs="unbounded" minOccurs="1" name="cpu" type="xs:float"/>

      <xs:element maxOccurs="unbounded" minOccurs="1" name="maxCPU"
    type="xs:nonNegativeInteger"/>
      <xs:element maxOccurs="unbounded" minOccurs="1"
    name="ifBandwidth" type="xs:nonNegativeInteger"/>
      <xs:element maxOccurs="unbounded" minOccurs="1"
    name="maxIfBandwidth" type="xs:nonNegativeInteger"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="rp-list">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="availability" type="xs:boolean"/>
      <xs:element name="bandwidth" type="xs:nonNegativeInteger"/>
      <xs:element name="maxBandwidth" type="xs:nonNegativeInteger"/>
      <xs:element name="activeConn" type="xs:nonNegativeInteger"/>
      <xs:element name="maxActiveConn" type="xs:nonNegativeInteger"/>
      <xs:element name="nsMap" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="ns-match-type">
    <xs:restriction base="xs:string">
```

```
      <xs:enumeration value="domain"/>
      <xs:enumeration value="uri"/>
      <xs:enumeration value="both"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="ns-list">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="id" type="xs:short"/>
      <xs:element name="matchType" type="ns-match-type"/>
      <xs:element name="uriMatch" type="xs:string"/>
      <xs:element name="domainMatch" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="disk-stat">
    <xs:sequence>
      <xs:element name="prereadState" type="xs:integer"/>
      <xs:element name="diskUsage" type="xs:float"/>
    </xs:sequence>
    <xs:attribute name="tier" type="xs:string"/>
  </xs:complexType>
  <xs:complexType name="app-type-http">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="namespace"
    type="ns-list"/>
      <xs:element maxOccurs="1" minOccurs="1"
    name="cpuUsage" type="xs:double"/>
      <xs:element maxOccurs="1" minOccurs="1"
    name="ramUsage" type="xs:double"/>
      <xs:element maxOccurs="unbounded" minOccurs="1" name="disk"
    type="disk-stat"/>
      <xs:element maxOccurs="unbounded" minOccurs="1"
    name="resourcePool" type="rp-list"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="na-type">
    <xs:sequence>
      <xs:element name="http" type="app-type-http"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="loadSnapshot">
    <xs:complexType>
      <xs:sequence>
 <xs:element maxOccurs="1" name="system" type="system-type"/>
 <xs:element maxOccurs="1" minOccurs="0" name="vma" type="vma-type"/>
 <xs:element maxOccurs="1" name="na" type="na-type"/>
      </xs:sequence>
      <xs:attribute name="at" type="xs:dateTime" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

NOTE: When the prereadState value is -1, do not consider the disk usage value.

Table 23 on page 241 defines the global device and server level counters. MaxDiskIOPS, MaxHTTPRate, MaxTCPConnRate are informational values and are not used to limit the

resource utilization. These counters are included in the XML response to the external monitoring server.

Table 23: Global Device and Server-Level Counters

Element Name	Description	Element Type	Description
HostName	Hostname of device or server	string	Character strings in XML
CPU	Total CPU utilization (includes native and VM applications)	cpu-usage-type	Percentage CPU utilization
MaxCPU	Maximum number of logical CPUs configured	float	Floating point numerical value
Bandwidth	Bandwidth delivered by server	NonNegativeInteger	Integer containing only positive values
MaxBandwidth	Maximum bandwidth capacity configured	NonNegativeInteger	Integer containing only positive values
DiskIOPS	Number of disk I/O operations from HTTP application	NonNegativeInteger	Integer containing only positive values
MaxDiskIOPS	Maximum disk I/O operations configured	NonNegativeInteger	Integer containing only positive values
TCPConnRate	TCP connection rate from HTTP application	Integer	Integer containing only positive values
MaxTCPConnRate	Maximum TCP connection rate configured	Integer	Integer containing only positive values

Table 24 on page 241 defines the resource pool counters.

Table 24: Resource Pool-Level Counters

Element Name	Description	Element Type	Description
Availability	Resource pool service availability	boolean;	Specifies a true or false value; true: configured and active; false: configured and inactive
Bandwidth	Total bandwidth used by resource pool	NonNegativeInteger	Integer containing only positive values
MaxBandwidth	Maximum bandwidth capacity configured for resource pool	NonNegativeInteger	Integer containing only positive values
ActiveConn	Number of concurrent TCP connections in response to resource pool	NonNegativeInteger	Integer containing only positive values
MaxActiveConn	Maximum number of concurrent connections for resource pool	NonNegativeInteger	Integer containing only positive values

Related Documentation

- Load Feedback API on page 235
- Example Load Feedback XML Schema Output on page 242

Example Load Feedback XML Schema Output

The following is an example of the load feedback API XML schema file output.

```
<?xml version="1.0" encoding="UTF-8"?>
<loadSnapshot xmlns="http://www.juniper.net/mfc/schema/lf"
at="2011-12-18T12:35:46Z">
 <system>
  <cpuInfo>
   <cpu id="0" util="0.98"/>
   <cpu id="1" util="51.23"/>
   <cpu id="2" util="66.78"/>
   <cpu id="3" util="75.06"/>
   <cpu id="4" util="80.24"/>
   <cpu id="5" util="83.53"/>
   <cpu id="6" util="95.76"/>
   <cpu id="7" util="87.52"/>
   <cpu id="8" util="88.80"/>
   <cpu id="9" util="90.23"/>
   <cpu id="10" util="90.93"/>
   <cpu id="11" util="91.85"/>
   <cpu id="12" util="92.41"/>
   <cpu id="13" util="93.30"/>
   <cpu id="14" util="100.00"/>
   <cpu id="15" util="93.83"/>
  </cpuInfo>
  <maxCPUUsage>100</maxCPUUsage>
  <ifBandwidth>53889351</ifBandwidth>
  <maxIfBandwidth>125000000</maxIfBandwidth>
  <diskInfo>
   <disk id="sda" util="4.36"/>
   <disk id="sdb" util="62.38"/>
   <disk id="sdc" util="41.09"/>
   <disk id="sdd" util="23.56"/>
   <disk id="sde" util="69.70"/>
   <disk id="sdf" util="32.57"/>
   <disk id="sdg" util="57.52"/>
   <disk id="sdh" util="46.04"/>
   <disk id="sdi" util="62.77"/>
   <disk id="sdj" util="95.05"/>
   <disk id="sdk" util="49.60"/>
   <disk id="sdl" util="69.41"/>
   <disk id="sdm" util="53.47"/>
   <disk id="sdn" util="44.65"/>
   <disk id="sdo" util="0.00"/>
   <disk id="sdp" util="0.00"/>
   <disk id="sdq" util="0.00"/>
  </diskInfo>
  <maxDiskUsage>100</maxDiskUsage>
  <httpTPS>0</httpTPS>
```

```
 <maxHttpTPS>200000</maxHttpTPS>
 <tcpConnRate>0</tcpConnRate>
 <maxTcpConnRate>0</maxTcpConnRate>
 <tcpActiveConn>1358</tcpActiveConn>
 <maxTcpActiveConn>64000</maxTcpActiveConn>
</system>
<na>
 <http>
  <cpuUsage>2.00</cpuUsage>
  <ramUsage>0.00</ramUsage>
  <disk tier="SSD">
   <prereadState>1</prereadState>
   <diskUsage>0.00</diskUsage>
  </disk>
  <disk tier="SAS">
   <prereadState>1</prereadState>
   <diskUsage>0.00</diskUsage>
  </disk>
  <disk tier="SATA">
   <prereadState>-1</prereadState>
   <diskUsage>0.00</diskUsage>
  </disk>
  <resourcePool>
   <name>global_pool</name>
   <availability>1</availability>
   <bandwidth>31117069</bandwidth>
   <maxBandwidth>1375000000</maxBandwidth>
   <activeConn>388</activeConn>
   <maxActiveConn>64000</maxActiveConn>
  </resourcePool>
 </http>
</na>
</loadSnapshot>
```

Related Documentation

- Load Feedback API on page 235
- Load Feedback API XML Schema Configuration on page 238

CHAPTER 10

Using Media Flow Controller Policy Engine

Policy Engine Overview

How the Policy Engine Works

The Policy Engine provides an infrastructure for operators to define rules that control the caching and delivery functions of Media Flow Controller. Media Flow Controller already provides many policy configuration options through its command-line interface (CLI) and Web interfaces. The Policy Engine provides a script-based configuration mechanism that is more comprehensive, more granular, and can be applied at runtime.

The following examples illustrate some of the capabilities of the Policy Engine:

- The Policy Engine provides a more granular set of configuration options such as the ability to override cache headers, control cache aging policies, and prevent unauthorized content downloads.
- A Media Flow Controller can use a policy to ingest multiple formats of the same video into the cache file system and then publish a single URL to the end user. When a request for content is received from a user, Media Flow Controller can redirect the user to the

appropriately formatted content based on the device (TV, PC, or Mobile) or player (Silverlight, Flash, or Quick Time) from which the user requests video.

- The Policy Engine allows an administrator to build rich sets of rules for controlling filtering behavior.

The Policy Engine is a flexible configuration mechanism that allows administrators more complete control over how Media Flow Controller manages content delivery so that it can easily adapt to rapidly changing needs.

What Is a Policy?

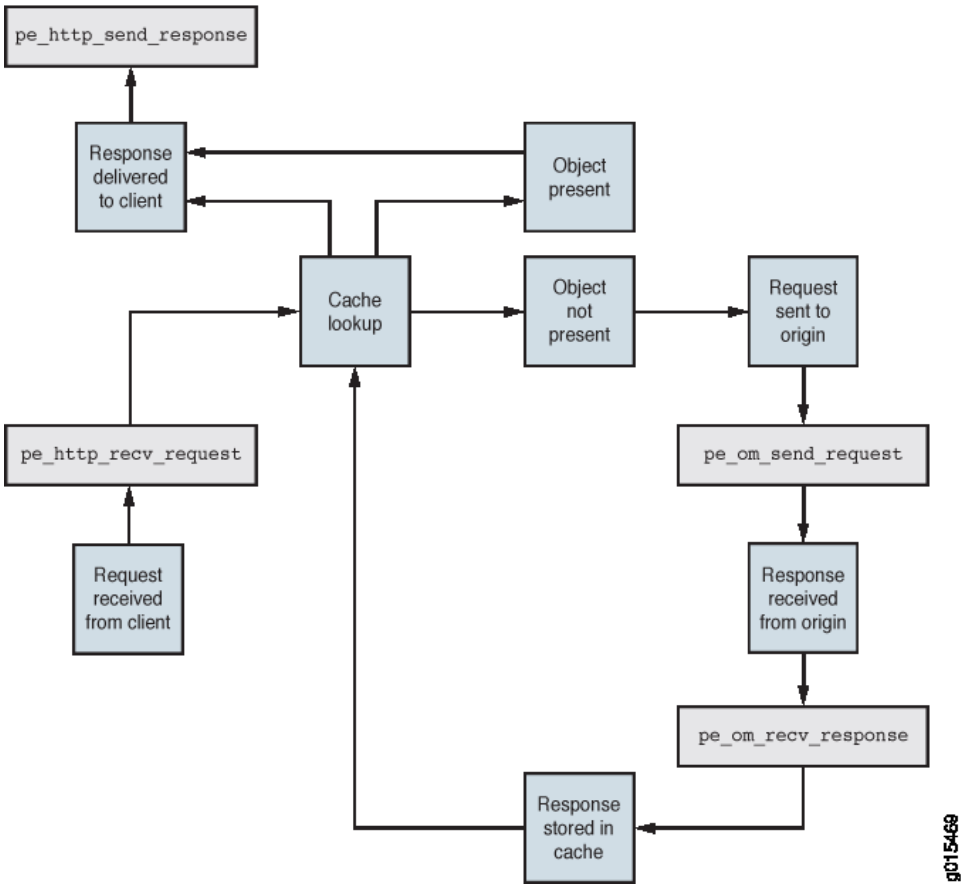
A ***policy*** is set of rules defined by an administrator. Policies are expressed as a Tool Command Language (TCL) script that is loaded onto a target Media Flow Controller, committed, and then bound to a single namespace with CLI commands.

Figure 15 on page 247 shows the relationship among a namespace, a policy script, policy procedures, and policies. Each namespace can have a single policy script bound to it. Each policy script must contain all the four predefined procedures, even though there is no rule defined in a procedure:

```
proc pe_http_recv_request {} {

}

proc pe_http_send_response {} {

}

proc pe_om_send_request {} {

}

proc pe_om_recv_response {} {

}
```

Figure 17 on page 249 describes how policy process names indicate **when** a procedure is executed. Finally, each procedure can have one or more policies.

Figure 15: Relationship Among Namespaces, Policy Script Files, and Policies



A policy rule is composed of conditions and actions. A **condition** is an expression in a TCL script. Here are some examples of conditions:

- If a request contains a cookie header
- If a client sends the request from the network A.B.C.D

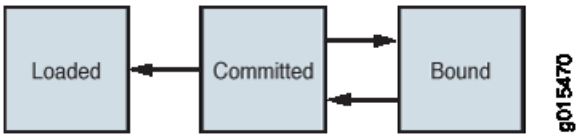
An **action** is taken as the result of a condition evaluation. Here are some examples of actions:

- Reject a request
- Cache or not cache an object
- Insert a header

Policy Life Cycle

Policies go through a simple life cycle described in Figure 16 on page 248.

Figure 16: Policy Life Cycle



The Policy Engine life cycle stages are:

- A policy is **loaded** when its script file is placed onto the file system of the target Media Flow Controller with either of the following commands:
  - **policy new** ***policy_file***
  - **policy fetch** ***policy_file http or scp URL of the remote machine and path of the policy script file***

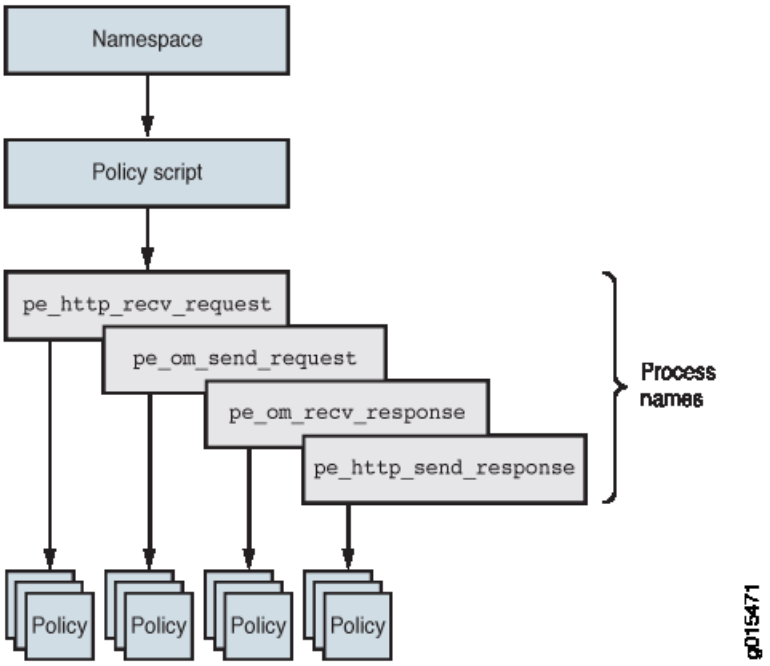
    < >
- A policy is **committed** when its syntax has been checked and it has been loaded into memory on the target Media Flow Controller with the **policy commit** ***policy_file*** CLI command.
- A policy is **bound** with a namespace with the **namespace** CLI command so that it can be applied to incoming and outgoing requests. Policies can be bound or unbound from a namespace without making the namespace status **inactive**.

Once a policy script file has been committed, the policy is stored in memory. A committed policy can only be bound to a single namespace. See "Loading, Committing, and Binding a Policy Script" on page 253" for an example of how to load, commit, and bind a policy script.

Policy Engine Execution

The Policy Engine provides a few well-defined insertion points for inserting policies into the operation of the Media Flow Controller. For example, when Media Flow Controller receives a request from a client, the Policy Engine provides an insertion point where a policy can be applied. Figure 17 on page 249 shows how these process procedures are executed at different stages in Media Flow Controller operation.

Figure 17: Policy Engine Process Procedure Names



Each of these insertion points correspond to a specific TCL **process procedure**. These procedures are described in Table 27 on page 257. These process procedures are registered in the Policy Engine when a given policy is bound to a namespace and executed when Media Flow Controller reaches those stages of operation.

You can replace a policy script file (Policy A) with a new policy script file (Policy B). We recommend that you change policies only when there are no active user sessions, such as during a maintenance period, to ensure that requests are treated with the correct policy.

This is recommended to avoid requests that are being serviced by the Media Flow Controller being treated with a subset of policies from Policy file A and a subset of policies from Policy file B, which could happen because each callout point in the policy script file is treated independently. So, if you do not change policies when users are logged off, an old policy might be applied when handling the response from the origin file (pe_om_recv_response invoked from Policy file A) and a new policy would be applied before sending the response to the client (pe_http_send_response invoked from Policy file B).

Building a Policy

The example policy below shows the mechanics of how a policy is constructed. The policy has a single condition and a single action. The TCL procedure name used in this sample script is taken from the list of process names described in Table 27 on page 257. The Policy Engine uses process names to indicate **when the policy is invoked**, not what the policy does. For this reason, it is necessary to use one of the process names described in Table 27 on page 257.

```
proc pe_http_recv_request {} {
  if [string match "10.*" [getparam socket.source_ip]] {
  setaction "reject_request";
    return;
  }
  return;
}
```

In this case, the process name **pe_http_recv_request** indicates that the policy runs after an HTTP request has been received and parsed. The procedure uses TCL string matching to isolate requests coming from the clients with IP addresses that begin with 10.x.x.x. When a request from such an IP address is found, the TCL procedure uses the **setaction** library procedure to return the action result. In this case, the action is **reject_request**. The request from other clients with IP addresses that do not begin with 10.x.x.x are processed normally.

Table 27 on page 257 describes several process names. Since this example script includes only a single procedure for the **pe_http_recv_request** process name, the Policy Engine uses default procedures with empty actions for the remaining process names. Here are the default procedure definitions:

```
proc pe_http_recv_request {} {}
proc pe_http_send_response {} {}
proc pe_om_send_request {} {}
proc pe_om_recv_response {} {}
```

The procedure below contains two policies:

```
proc pe_om_recv_response {} {
  set uri [getparam http.req.uri];
  if [string match *.mpg $uri] {
    setaction "cache_object";
    return;
  }
  if [regexp {vod[0-9].*mp4} $uri] {
    setaction "no_cache_object";
    return;
  }
  return;
}
```

This procedure uses the process name **pe_om_recv_response**, which matches HTTP received responses from the origin server. The first line sets a local variable based on the value of a Policy Engine **token**. These are TCL global variables that contain useful Media Flow Controller runtime data. Tokens are described in Table 28 on page 258. The example above uses **http.req.uri**, which provides a string representation of the requested URI.

See "Policy Examples Based on pe_http_recv_request" on page 269 and "Policy Examples Based on pe_om_recv_response" on page 276 for a list of useful policy script examples.

Related Documentation

- Media Flow Controller Policy Engine on page 17
- Policy Engine Errors and Debugging on page 262
- Policy Engine API Reference on page 257

Creating and Editing Policy Script Files

A policy script file can be created with any text editor on another system and then copied onto a Media Flow Controller with the **policy fetch** CLI command. Or it can be created directly on the target Media Flow Controller with the **policy new** CLI command.

Once created, policy script files can be edited on the target Media Flow Controller with the **policy edit** CLI command. The **policy new** and **policy edit** commands both launch the **vi** text editor.

NOTE: Media Flow Controller might not support all the available TCL commands within the Policy Engine script. Be careful when using commands that are not used in the examples in the "Policy Examples Based on pe_om_recv_response" on page 276 and "Policy Examples Based on pe_http_recv_request" on page 269 sections.

Editing a Policy Script File with vi

vi is a text editor commonly available on Linux systems. See *Learning the vi and Vim Text Editors* (O'Reilly Media, 2008) for a complete introduction to vi. The following sections provide a quick overview to common vi tasks.

vi Modes

vi uses three common modes: **keyboard command mode**, **text entry mode**, and **bottom-line mode** . When vi first launches, it is in command mode which allows a user to move the cursor, and add and delete text. The cursor (usually an underscore) is the center of text editing. Table 25 on page 251 describes common keyboard command mode operations.

Table 25: vi Keyboard Command Mode Operations

Operation	Description
h	Move cursor backwards.
j	Move cursor down.
k	Move cursor up.
l	Move cursor forward.

Table 25: vi Keyboard Command Mode Operations *(continued)*

Operation	Description
<SPACE>	Move cursor forward.
<RETURN>	Move cursor down.
w	Move cursor one word.
x	Delete the current character.
dd	Delete the current line.
dw	Delete the current word.
i	Insert at the current cursor position.
o	Open a line below the current cursor position.

Entering Text with vi

Text entry with vi is a three-stage process:

1. Move the cursor to a place in the file that needs editing with one of the keyboard navigation operations.
2. Begin text entry mode with one of the text entry keyboard operations: **i** or **o**.
3. Enter the necessary text.
4. Return to keyboard command mode with the **ESC** key.

Replacing Text With vi

Editing text with vi is a three-stage process:

1. Move the cursor to a place in the file that needs editing with one of the keyboard navigation operations.
2. Delete some text, if necessary.
3. Begin text entry mode with one of the text entry keyboard operations: **i** or **o**.
4. Enter the necessary text.
5. Return to keyboard command mode with the **ESC** key.

Saving and Quitting vi

After text editing is complete, you can save your work to a file. Each file has a filename which should be descriptive of its purpose. The command sequence to save a file is based on the bottom-line mode.

1. Begin with the keyboard command mode.
2. Press the colon (:) key to enter bottom line mode.
3. Type **w** followed by the filename.
4. Type **:quit** to exit vi.

Related Documentation

- Loading, Committing, and Binding a Policy Script on page 253
- Providing Geographical Location-Based Services with Policies on page 254
- Creating and Editing Policy Script Files on page 251
- Policy Engine Overview on page 245

Loading, Committing, and Binding a Policy Script

To load, commit, and bind a policy script:

1. Create the text file for the policy script file.
2. Load the policy script file onto the target Media Flow Controller.

   ```
   MFC (config)# policy fetch MyPolicy scp//admin:admin@myhost/home/admin/
     MyPolicy.tcl
   ```

3. Commit the policy.

   ```
   MFC (config)# policy commit MyPolicy
   ```

   Media Flow Controller checks the syntax of the TCL script and loads it into the Policy Engine.

4. Bind the policy to a namespace.

   ```
   MFC (config)# namespace danspace bind policy MyPolicy
   ```

   At this point the policy is bound and operational.

5. Unbind the policy.

   ```
   MFC (config)# no namespace danspace bind policy MyPolicy
   ```

   At this point the policy is committed but not bound.

Policy command-line interface (CLI) commands are described in the *Media Flow Controller CLI Command Reference***policy** section.

Related Documentation

- Policy Engine GeoIP Information Lookup on page 256
- Creating and Editing Policy Script Files on page 251
- Providing Geographical Location-Based Services with Policies on page 254
- Policy Engine Overview on page 245

Providing Geographical Location-Based Services with Policies

Providing Geographical Location-based Services with Policies Overview

Policies can use geographical information to control policy execution in the **pe_http_recv_request** procedure. For example, a policy script can select a country-specific origin server.

Media Flow Controller can detect the geographical location of a user based on the user's IP address. CDNs or content providers can use the geographical information of the user to offer various services. Here are some example use cases:

- Location-based service differentiation:
  - Serve different objects to different users based on user location
  - Allow service providers to serve English content to users accessing it from the USA and Chinese content for users accessing it from China.
  - High quality content for select locations and standard quality content for others (differentiated based on bit-rate profile information in the filename).
- Redirect a user to a friendly error page when specific content cannot be served due to Geo blocking.
- Redirect a user to the nearest edge server or cache based on GeoIP information. For example, a user from Asia directly accessing content from origin servers in USA would be redirected to edge caches in Asia.
- Allow an edge cache to decide which origin server or cache to go based on a user's GeoIP information. For example, a media library may be partitioned and replicated across several origin servers or caches. Based on a user's GeoIP information, an edge cache can go to the appropriate origin server/caches to fetch content.

Installing the MaxMind GeoIP Database

Geographical information lookup is based on information retrieved from the MaxMind GeoIP database, which must be installed before configuring geographical information lookup with Media Flow Controller. Media Flow Controller supports several GeoIP databases, but each database should be downloaded and installed separately. Media Flow Controller supports the following database download or install scenarios:

- GeoIPCity.dat and GeoIPISP.dat
- GeoIPCountry.dat and GeoIPISP.dat

You can install either the GeoIPCity.dat or GeoIPCountry.dat database but not both, as these databases are mutually exclusive. In addition, GeoIPISP.dat must be installed after you install GeoIPCity.data or GeoIPCountry.dat.

To download and install the MaxMind GeoIP database:

1. Download the GeoIP database from the MaxMind website (www.maxmind.com). The download file should be in **.tar** format.
2. Untar the download file onto a specific location that can be accessed from the target Media Flow Controller device.

NOTE: **The filename should be either .dat or .dat.gz.**

3. Download the **socket.source_portuntarred .dat** or **.dat.gz** file onto the target Media Flow Controller device.

   application maxmind download <*filename*>

4. Install the GeoIP database.

   application maxmind install <*filename*>

Configuring Geographical Information Lookup

Configure geographical information lookup information using the following CLI command:

namespace *name* delivery protocol http client-request geo-service maxmind

Use the following CLI to define the behavior when the geographical data lookup service is unavailable. By default the user connection is rejected if the service is unavailable. If bypass is enabled, Media Flow Controller continues processing the request even if the service is unavailable.

namespace *name* delivery protocol http client-request geo-service failover-mode bypass

Example: Geographical Information Lookup

Once enabled, the lookup is performed after parsing the HTTP request header with the **pe_http_recv_request** procedure. Geographical data tokens are described in Table 29 on page 259. For example:

```
proc pe_http_recv_request {} {
  set resp [getparam socket.geodata.country_code];
  if [string match "US" $resp] {
    setaction set_origin_server "us.test.com";
    return;
  }
}
```

Related Documentation

- Policy Engine GeoIP Information Lookup on page 256
- Creating and Editing Policy Script Files on page 251
- Providing Geographical Location-Based Services with Policies on page 254

- Policy Engine Overview on page 245

Policy Engine GeoIP Information Lookup

Media Flow Controller supports GeoIP services for HTTP delivery. The Policy Engine MaxMind GeoIP database provides valuable geographical information about Internet visitors, such as country, state, city, and postal code, for policy decisions. Upon receiving a client request, Media Flow Controller performs GeoIP database lookup and extracts all the details (such as country, state, city, postal code, and so on) about the client IP address. Based on the configured policy, a client is denied access or redirected to a different Web page.

For example, a CDN service provide might have rights to serve media only to select geographical locations. The service blocks access to content for clients accessing it from unauthorized geographical locations. Alternatively, a client is redirected to an error page.

The geographical information is available as tokens inside the Policy Engine.

- Configuring GeoIP Lookup on page 256
- Using GeoIP Lookup Tokens on page 256
- Using GeoIP Lookup Response Codes on page 256

Configuring GeoIP Lookup

To configure GeoIP lookup on the Media Flow Controller:

1. Download and install the MaxMind GeoIP database.

   application maxmind download *filename*
   application maxmind install *filename*

2. Create and save a policy script file. Apply this policy to a namespace by loading the policy script on the Media Flow Controller, committing the policy, and binding the policy to a namespace.

   policy fetch *policy-name* **scp://***username***:***password***@***hostname/pathname*
   policy commit *policy-name*
   namespace *name* **bind policy** *policy-name*

3. Configure the namespace for GeoIP lookup.

   namespace *name* **delivery protocol http client-request geo-service maxmind**

Using GeoIP Lookup Tokens

Once enabled, the lookup occurs after parsing the HTTP request header. Geographical information tokens are available, as defined in Table 29 on page 259.

Using GeoIP Lookup Response Codes

Policy Engine provides response codes for Geo IP lookup, as described in Table 26 on page 257.

Table 26: GeoIP Response Codes

Response Code	Action
200 - OK	GeoIP lookup was successful.
404	IP address was not found in the Geo database.
500	Geo database lookup error.

Related Documentation

- Creating and Editing Policy Script Files on page 251
- Providing Geographical Location-Based Services with Policies on page 254
- Policy Engine Overview on page 245

Policy Engine API Reference

Table 27: Policy Engine Process Procedures

Name	Supported Tokens	Supported Actions	Description
pe_http_recv_request	socket.source_ip socket.source_port socket.dest_ip socket.dest_port http.req.host http.req.uri http.req.cookie http.req.header http.req.method http.req.query	reject_request redirect cache_object no_cache_object set_origin_server url_rewrite insert_header modify_header remove_header cache_index_append transmit_rate	Executed when an HTTP request has been received and parsed.
pe_http_send_response	http.req.host http.req.uri http.req.cookie http.req.header http.req.method http.req.query http.res.status_code http.res.header	insert_header remove_header modify_header set_ip_tos	Executed right before an HTTP response is returned.
pe_om_send_request	socket.source_ip socket.source_port socket.dest_ip socket.dest_port http.req.host http.req.uri http.req.cookie http.req.header http.req.method http.req.query	insert_header remove_header modify_header set_ip_tos	Executed right before an HTTP request is forwarded to the origin manager. NOTE: If the request processed by Media Flow Controller gets tunneled, then the Policy Engine won't evaluate the *pe_om_send_request* and *pe_om_recv_response* insertion points of the Policy Engine script.

Table 27: Policy Engine Process Procedures *(continued)*

Name	Supported Tokens	Supported Actions	Description
pe_om_recv_response	socket.source_ip socket.source_port socket.dest_ip socket.dest_port http.req.host http.req.uri http.req.cookie http.req.header http.req.method http.req.query om.res.status_code om.res.header	cache_object no_cache_object redirect insert_header modify_header remove_header	Executed when the origin manager receives and parses an HTTP response. NOTE: If the request processed by Media Flow Controller gets tunneled, then the Policy Engine won't evaluate the *pe_om_send_request* and *pe_om_recv_response* insertion points of the Policy Engine script.

Table 28: Policy Engine Tokens

Group	Token	Description
Global Socket Level	socket.source_ip	A socket represents the connection between the client and the Media Flow Controller or the connection between the Media Flow Controller and the origin server. Each connection will have a *source_ip*, *source_port*, *dest_ip*, and *dest_port*. The source is the device that originated the connection and the destination is the device that terminates the connection. In the case of Policy Engine, *source_ip* represents the client's IP address and *source_port* represents its port number. In case of the reverse proxy deployment, *dest_ip* represents Media Flow Controller's IP and *dest_port* represents its port number. In the case of transparent proxy deployment, *dest_ip* can represent Media Flow Controller's IP or the actual origin server's IP address, and *dest_port* represents its port number.
	socket.source_port	
	socket.dest_ip	
	socket.dest_port	
Client Request Object	http.req.host	Returns the Host header's value in the request.
	http.req.cookie *cookie*	Returns the value of the specified cookie name and its value from the request's cookie header.
	http.req.header *header*	Returns the value of the specified header field. Header can be a standard HTTP 1.0 or 1.1 header field or a customized HTTP (X-header) header field. If multiple headers exist with the same name, only the value of the first is returned.
	http.req.method	HTTP methods (GET, POST, and so on).
	http.req.query	Returns all the query parameters (starts with the character " ?").
Origin Server Response Object	om.res.status_code	Response code from the origin server.
	om.res.header *header*	Return the value of the specified header or customized HTTP header. If multiple headers exist with the same name, only the value of the first is returned.

Table 29: Geographical Data Tokens

Token	Description
socket.geodata.status_code	Returns the status code returned by the geographical information lookup.
socket.geodata.continent_code	Returns the two-letter continent code.
socket.geodata.country_code	Returns the two-letter country code (ISO 3166-1).
socket.geodata.country	Returns the country.
socket.geodata.state	Returns the state.
socket.geodata.city	Returns the city.
socket.geodata.postal_code	Returns the postal code.
socket.geodata.isp	Returns ISP information, if available.

Table 30: Policy Engine Actions

Group	Action	Description
pe_http_recv_request	**reject_request [geoip]**	Return "403 Forbidden 54001" to client. 54001 indicates that the request was rejected by a defined policy. If optional parameter "geoip" is passed, sub-error code is set as 54102. The sub-error code can be logged in access logs.
	url_rewrite ***host uri***	Use this new URI instead of the one in the request. If host or URI is empty, it is not used. If host is changed, it would be used only in follow header host configuration. Use the new URI to fetch from the origin server and cache the object in the new URI name. The access log logs the original request URI. The new URI is logged by specifying **i{X-NKN-Remapped-Uri}** in the access log format. Note: Do not use absolute URL in the url_rewrite action.
	no_cache_object (GET only)	Do not cache the object.
	cache_object (GET only)	Cache the object. If the Policy Engine caches an object, but the origin manager returns " Cache-control: no-cache" , then Media Flow Controller overwrites the origin directive and caches the object.
	set_origin_server ***domain*****:** ***port***	Use this origin server instead of the one configured in the namespace.
	insert_header ***name value***	Insert the given header to the request headers. If the given header is already present, the header is not inserted.
	modify_header ***name value***	Modify the given header in the request headers. If the given header is already present, the header is changed to the new value or the value is inserted as new header.
	remove_header ***name***	Remove the given header from the request headers.
	redirect ***uri*** **[geoip]**	Sends a redirect response to the client with the new location. If the URL redirection happens due to policy, a suberror code is set to 54002. If an optional parameter "geoip" is passed, the sub-error code is set to 54103. The sub-error code can be logged in access logs.
	cache_index_append	Appends the given string to the cache index name. The cache index is used internally to identify unique objects in the cache. Request URI and the request sent to the origin are not changed.
	transmit_rate	

Table 30: Policy Engine Actions *(continued)*

Group	Action	Description
		Customizes the bandwidth settings for a per-user connection (as opposed to the virtual player settings, which are applicable to all the users that are mapped to a given namespace). It specifies the bandwidth settings for this object. It accepts three parameters: • Maximum bandwidth—Value range: 0-24 Mbps. • Maximum bit rate for the video—Value range: 0-16 Mbps. • Fast start time/size—Any non-negative value. The first two parameters are in bits per second; however, these parameters could be in Gbps, Mbps, Kbps, or bps. The last parameter is in bytes or seconds. The allowed units are MB, KB, or seconds. If any parameter is not set, the null string ("") is passed.
pe_http_send_response	insert_header *name value*	Insert the given header to the response headers. If the given header is already present, the header is not inserted.
	modify_header *name value*	Modify the given header in the response headers. If the given header is already present, the header is changed to the new value else the value is inserted as new header.
	remove_header *name*	Remove the given header from the response headers.
	set_ip_tos 0b*tos*	Setup socket DSCP/TOS byte. The DSCP/TOS byte is configured as binary format. It should have six bits. The string is led by "0b". For example, 0b000111.
pe_om_send_request	insert_header *name value*	Insert the given header to the Origin request headers. If the given header is already present, the header is not inserted.
	modify_header *name value*	Modify the given header in the Origin request headers. If the given header is already present, the header is changed to the new value else the value is inserted as new header.
	remove_header *name*	Remove the given header from the Origin request headers.
	set_ip_tos 0b*tos*	Setup socket DSCP/TOS byte. The DSCP/TOS byte is configured as binary format. It should have six bits. The string is led by "0b". For example, 0b000111.

Table 30: Policy Engine Actions *(continued)*

Group	Action	Description
pe_om_recv_response	**cache_object**	Cache the object. If the Policy Engine tries to cache an object, but the origin manager returns "Cache-control: no-cache" , then Media Flow Controller overwrites the origin directive and caches the object.
	no_cache_object	Do not cache the object. Equivalent to "Cache-Control: no-cache".
	redirect *uri* **[geoip]**	Redirect the browser or player to a new location.
	insert_header *name value*	Insert a new header in the response headers. If the given header is already present, the header is not inserted.
	modify_header *name value*	Modify the given header in the response headers. If the given header is already present, the header is changed to the new value or the value is inserted as new header.
	remove_header *name*	Remove the given header from the response headers.

Table 31: Policy Engine Utility Procedures

Procedure	Description
getparam *token*	Get the value from the requested token.
setaction *action*	Perform the requested action.

Related Documentation

- Creating and Editing Policy Script Files on page 251
- Policy Engine Overview on page 245
- Policy Engine Errors and Debugging on page 262
- Media Flow Controller Policy Engine on page 17

Policy Engine Errors and Debugging

The Policy Engine has full support for TCL scripting features. When a script is committed, the Policy Engine runs it through a syntax checker. If the script passes the syntax checker, it is committed. Other error and debugging features include:

- Each TCL script parsing or evaluation error will generate an error message that is then added to a namespace-specific error log.
- The Policy Engine uses different counters for each TCL script parsing or evaluation error. The counter name pattern is "glob_pe_error_**xxxx**," described in Table 32 on page 263.

NOTE: You can view some of these Policy Engine error counters by entering the **show counters** command in the CLI. However, many of these error counters cannot be readily viewed using the CLI and are only accessible for debugging purposes, which requires guidance from technical support personnel at the Juniper Networks Technical Assistance Center (JTAC).

Table 32: Policy Engine Error Counters

Category	Counter	Description
Policy evaluation counters	glob_pe_eval_http_recv_req	Number of times HTTP receive policy was evaluated.
	glob_pe_eval_om_recv_resp	Number of times origin receive response policy was evaluated.
	glob_pe_eval_om_send_req	Number of times origin send policy was evaluated.
	glob_pe_eval_err_http_recv_req	Number of times evaluation of HTTP receive policy failed.
	glob_pe_eval_err_om_recv_resp	Number of times evaluation of origin response policy failed.
	glob_pe_eval_err_om_send_req	Number of times evaluation of origin send policy failed.
	glob_pe_eval_http_send_resp	Number of times HTTP send response policy was evaluated.
	glob_pe_eval_err_http_send_resp	Number of times evaluation of HTTP send response policy failed.
Action Counters for HTTP Requests	glob_pe_act_http_req_cache_object	Number of times action cache_object was set.
	glob_pe_act_http_req_no_cache_object	Number of times action no_cache_object was set.
	glob_pe_act_http_req_redirect	Number of times action redirect was set.
	glob_pe_act_http_req_reject_request	Number of times action reject_request was set.
	glob_pe_act_http_req_set_origin_server	Number of times action set_origin_server was set.
	glob_pe_act_http_req_url_rewrite	Number of times url_rewrite action was set by policy script in pe_http_recv_request procedure.

Table 32: Policy Engine Error Counters *(continued)*

Category	Counter	Description
	glob_pe_act_http_req_insert_header	Number of times insert_header action was set by policy script in pe_http_recv_request procedure.
	glob_pe_act_http_req_modify_header	Number of times modify_header action was set by policy script in pe_http_recv_request procedure.
	glob_pe_act_http_req_remove_header	Number of times remove_header action was set by policy script in pe_http_recv_request procedure.
Action Counters for Origin Response	glob_pe_act_om_rsp_cache_object	Number of times action cache_object was set by policy script in pe_om_recv_response procedure.
	glob_pe_act_om_rsp_no_cache_object	Number of times action no_cache_object was set by policy script in pe_om_recv_response procedure.
	glob_pe_act_om_rsp_redirect	Number of times action redirect was set by policy script in pe_om_recv_response procedure.
	glob_pe_act_om_rsp_insert_header	Number of times action insert_header was set by policy script in pe_om_recv_response procedure.
	glob_pe_act_om_rsp_remove_header	Number of times action remove_header was set by policy script in pe_om_recv_response procedure.
	glob_pe_act_om_rsp_modify_header	Number of times action modify_header was set by policy script in pe_om_recv_response procedure.

Table 32: Policy Engine Error Counters *(continued)*

Category	Counter	Description
Action Counters for Origin Request	glob_pe_act_om_req_modify_header	Number of times action modify_header was set by policy script in pe_om_send_request procedure.
	glob_pe_act_om_req_remove_header	Number of times action remove_header was set by policy script in pe_om_send_request procedure.
	glob_pe_act_om_req_insert_header	Number of times action insert_header was set by policy script in pe_om_send_request procedure.
	glob_pe_act_om_req_set_ip_tos	Number of times action set_ip_tos was set by policy script in pe_om_send_request procedure.
pe_http_recv_request Parser Error Counters	glob_pe_err_http_req_get_no_host	Unable to get a host header.
	glob_pe_err_http_req_get_no_method	Unable to get a method.
	glob_pe_err_http_req_get_no_token	getparam called without a token.
	glob_pe_err_http_req_get_no_uri	Unable to get a URI.
	glob_pe_err_http_req_get_unknown_token	getparam called with an unsupported token.
	glob_pe_err_http_req_set_action_cache_index_append	Set when Policy Engine fails to set the cache index with the given pattern.
	glob_pe_err_http_req_set_action_transmit_rate	Set when Policy Engine fails to set the transmit-rate parameters.
	glob_pe_err_http_req_set_no_cache_object	setaction no_cache_object called for non GET method.
	glob_pe_err_http_req_set_no_action	setaction called without an action parameter.
	glob_pe_err_http_req_set_origin_server	setaction set_origin_server called without a parameter.
	glob_pe_err_http_req_set_redirect	setaction redirect called without a parameter.
	glob_pe_err_http_req_set_unknown_action	setaction called with an unsupported action.

Table 32: Policy Engine Error Counters *(continued)*

Category	Counter	Description
pe_om_send_request Parser Error Counters	glob_pe_err_om_req_get_no_host	Unable to get a host header.
	glob_pe_err_om_req_get_no_method	Unable to get a method.
	glob_pe_err_om_req_get_no_token	getparam called without a token.
	glob_pe_err_om_req_get_no_uri	Unable to get URI.
	glob_pe_err_om_req_get_unknown_token	getparam called with an unsupported token.
	glob_pe_err_om_req_set_ins_hdr	Unable to insert the header.
	glob_pe_err_om_req_set_ins_hdr_param	setaction insert_header called without required parameters.
	glob_pe_err_om_req_set_mod_hdr	Unable to modify a header.
	glob_pe_err_om_req_set_mod_hdr_param	setaction modify_header called without required parameters.
	glob_pe_err_om_req_set_no_action	setaction called without an action parameter.
	glob_pe_err_om_req_set_rmv_hdr	Unable to remove a header.
	glob_pe_err_om_req_set_rmv_hdr_param	setaction remove_header called without parameter.
	glob_pe_err_om_req_set_set_ip_tos	setaction set_ip_tos called without a parameter or a wrong parameter.
	glob_pe_err_om_req_set_unknown_action	setaction called with an unsupported action.

Table 32: Policy Engine Error Counters *(continued)*

Category	Counter	Description
pe_om_recv_response Parser Error Counters	glob_pe_err_om_rsp_get_no_host	Unable to get a host header.
	glob_pe_err_om_rsp_get_no_method	Unable to get a method.
	glob_pe_err_om_rsp_get_no_token	getparam called without a token.
	glob_pe_err_om_rsp_get_no_uri	Unable to get a URI.
	glob_pe_err_om_rsp_get_unknown_token	getparam called with unsupported token.
	glob_pe_err_om_rsp_set_ins_hdr	Unable to insert the header.
	glob_pe_err_om_rsp_set_ins_hdr_param	setaction insert_header called without required parameters.
	glob_pe_err_om_rsp_set_no_action	setaction called without an action parameter.
	glob_pe_err_om_rsp_set_redirect	setaction redirect called without a parameter.
	glob_pe_err_om_rsp_set_rmv_hdr	Unable to remove a header.
	glob_pe_err_om_rsp_set_rmv_hdr_param	setaction remove_header called without parameter.
	glob_pe_err_om_rsp_set_set_ip_tos setaction	set_ip_tos called without a parameter or a wrong parameter.
	glob_pe_err_om_rsp_set_unknown_action	setaction called with an unsupported action.

Table 32: Policy Engine Error Counters *(continued)*

Category	Counter	Description
pe_http_send_response Parser Error Counters	glob_pe_err_http_rsp_get_no_host	Unable to get host header.
	glob_pe_err_http_rsp_get_no_method	Unable to get method.
	glob_pe_err_http_rsp_get_no_token	getparam called without token.
	glob_pe_err_http_rsp_get_no_uri	Unable to get URI.
	glob_pe_err_http_rsp_get_unknown_token	getparam called with unsupported token.
	glob_pe_wrn_http_rsp_get_no_rsp_header	Given response header was not found.
	glob_pe_wrn_http_rsp_get_no_cookie	Given cookie was not found.
	glob_pe_wrn_http_rsp_get_no_header	Given header was not found.
	glob_pe_wrn_http_rsp_get_no_query	No query for this URI.
	glob_pe_err_http_rsp_set_ins_hdr	Unable to insert the header.
	glob_pe_err_http_rsp_set_ins_hdr_param	setaction insert_header called without required parameters.
	glob_pe_err_http_rsp_set_mod_hdr	Unable to modify header.
	glob_pe_err_http_rsp_set_mod_hdr_param	setaction modify_header called without parameter.
	glob_pe_err_http_rsp_set_no_action	setaction called without action parameter.
	glob_pe_err_http_rsp_set_rmv_hdr_param	Unable to remove header.
	glob_pe_err_http_rsp_set_rmv_hdr	setaction remove_header called without parameter.
	glob_pe_err_http_rsp_set_set_ip_tos	setaction set_ip_tos called without parameter or wrong parameter.
	glob_pe_err_http_rsp_set_unknown_action	setaction called with unsupported action.

Table 32: Policy Engine Error Counters *(continued)*

Category	Counter	Description
Action Counters for HTTP Response	glob_pe_act_http_rsp_insert_header	Number of times action insert_header was set by policy script in pe_http_send_response procedure.
	glob_pe_act_http_rsp_remove_header	Number of times action remove_header was set by policy script in pe_http_send_response procedure.
	glob_pe_act_http_rsp_modify_header	Number of times action modify_header was set by policy script in pe_http_send_response procedure.
	glob_pe_act_http_rsp_set_ip_tos	Number of times action set_ip_tos was set by policy script in pe_http_send_response procedure.

Related Documentation

- Creating and Editing Policy Script Files on page 251
- Policy Engine Overview on page 245
- Policy Engine API Reference on page 257
- Media Flow Controller Policy Engine on page 17

Policy Examples Overview

Policy Examples Based on pe_http_recv_request

The examples in the following sub-sections illustrate policy scripts based on the pe_http_recv_request.

Setting an Origin Server

The examples below show how to set an origin server. The first example uses a URI pattern to determine which origin server to use.

```
proc pe_http_recv_request {} {
  set uri [getparam http.req.uri];
  if [string match "*.jpg" $uri] {
   setaction "set_origin_server" "jpg.origin.com";
  } elseif [string match "*.mp3" $uri] {
   setaction "set_origin_server" "mp3.origin.com";
  } elseif [string match "*.gif" $uri] {
   setaction "set_origin_server" "gif.origin.com";
  } else {
   setaction "set_origin_server" "others.origin.com";
  }
  return;
}
```

The second example uses an origin server with a non-default HTTP port.

```
proc pe_http_recv_request {} {
  set uri [getparam http.req.uri];
  if [string match "*.jpg" $uri] {
   setaction "set_origin_server" "jpg.origin.com:8080";
   return;
  } else {
   setaction "set_origin_server" "others.origin.com:8081";
   return;
  }
  return;
}
```

Blocking Requests Based on IP Addresses

The example below shows how to block requests based on IP addresses.

```
proc pe_http_recv_request {} {
  if [string match "10.*" [getparam socket.source_ip]] {
   setaction "reject_request";
   return;
  }
  return;
}
```

Blocking Requests Based on Regular Expressions

The first example shows how to block requests based on regular expressions.

```
proc pe_http_recv_request {} {
  set host [getparam http.req.host];
    if { [regexp {video[0-9]*.youtube.com} $host] } {
      setaction "reject_request";
   return;
  }
  return;
}
```

The second example shows how to block requests from clients with IP addresses from **107.xx.xx.xx** or **121.xx.xx.xx**.

```
proc pe_http_recv_request {} {
  set src_ip [getparam socket.source_ip];
  if { [string match "107.*" $src_ip] == 1 } {
   setaction "reject_request";
   return;
  }
  if { [string match "121.*" $src_ip] == 1 } {
   setaction "reject_request";
   return;
  }
  return;
}
```

The third example shows how to reject all URIs for **/foo/*.gif**.

```
proc pe_http_recv_request {} {
  if { [ regexp ".*\/foo\/.*\.gif$" [getparam http.req.uri]] == 1 } {
   setaction "reject_request";
   return;
  }
  return;
}
```

Blocking User Requests from Certain Geographical Locations

The example below shows how to block requests from certain geographical locations.

```
proc pe_http_recv_request {} {
  if {[string match "10.24.*" [getparam socket.source_ip]] || [string match
"172.12.32.*" [getparam socket.source_ip]]} {
   setaction "reject_request";
   return;
```

```
  }
  return;
}
```

Blocking User Requests Based on URL-Based User Location

The example below shows how to block user requests based on user location information contained in a URL.

```
proc pe_http_recv_request {} {
  if [string match "*location=China*" [getparam http.req.query]] {
   setaction "reject_request";
   return;
  }
  return;
}
```

Blocking Malicious Users

The example below shows how to block malicious users.

```
proc pe_http_recv_request {} {
  if [string match "10.24.1.4" [getparam socket.source_ip]] {
   setaction "reject_request";
   return;
  }
  return;
}
```

Blocking Users from Viewing Certain Content

The example below shows how to block users from viewing certain content.

```
proc pe_http_recv_request {} {
  if { [ string match "/images/*.jpg" [ getparam http.req.uri ] ] ||
    [ string match "/images/*.jepg " [getparam http.req.uri ] ] } {
   setaction "reject_request";
   return;
  }
  return;
}
```

Blocking Requests Based on the User's Platform

The example below shows how to block a request based on the user's platform (e.g., device, browser, or operating system).

```
proc pe_http_recv_request {} {
  set header [getparam http.req.header User-Agent];
  if {[string match "*Opera*" $header] == 1} {
   setaction "reject_request";
   return;
  }
  return;
}
```

Blocking Requests from Unauthorized Web Portals

The example below shows how to block requests from unauthorized web portals.

```
proc pe_http_recv_request {} {
  set header [getparam http.req.header Referer];
  if {[string match "*www.portal.tv*" $header] != 0} {
   setaction "reject_request";
   return;
  }
  return;
}
```

Redirecting Users to Different Websites Based on Their Platform

The first example below shows how to redirect users to different websites based on their platform.

```
proc pe_http_recv_request {} {
  set header [getparam http.req.header User-Agent];
  if {[string match "*iPhone*" $header] == 1} {
   setaction "redirect" "http://iphone.example.com/";
   return;
  }
  return;
}
```

The second example below shows how to redirect users with an unsupported device.

```
proc pe_http_recv_request {} {
  set header [getparam http.req.header User-Agent];
  if {[string match "*iPhone*" $header] == 0} {
   setaction "redirect" "http://www.example.com/sorry.html";
   return;
  }
  return;
}
```

Redirecting Unauthorized Viewers to a Registration Portal

The example below shows how to redirect unauthorized viewers to a registration portal.

```
proc pe_http_recv_request {} {
  set cookie [getparam http.req.cookie test_cookie];
  if {[string match *guest* $cookie]} {
   setaction "redirect" "http://register.myserver.com";
   return;
  }
  return;
}
```

Do Not Cache Content Based on the Object's Suffix

The example below shows how to avoid caching content based on the object's suffix.

```
proc pe_http_recv_request {} {
  if {[regexp {(.php|.cgi|.asp)} [getparam http.req.uri]]} {
  setaction "no_cache_object";
   return;
  }
```

```
  return;
}
```

Do Not Cache Content Based on a Cookie

The example below shows how to avoid caching content based on a cookie.

```
proc pe_http_recv_request {} {
  set cookie [getparam http.req.cookie test_cookie];
  if {[string bytelength $cookie] > 0} {
   setaction "no_cache_object";
   return;
  }
  return;
}
```

Masking Deployment Complexity

The example below shows how to mask deployment complexity.

```
proc pe_http_recv_request {} {
  set uri [getparam http.req.uri];
  if [string match "*.mpg" $uri] {
   setaction "set_origin_server" "images.myserver.com";
   return;
  }
  if [string match "*.html" $uri] {
   setaction "set_origin_server" "content.myserver.com";
   return;
  }
}
```

Distributing Load Across Multiple Origin Servers

The example below shows how to distribute load across multiple origin servers.

```
proc pe_http_recv_request {} {
  set uri [getparam http.req.uri];
  if [string match "*.jpg" $uri] {
   setaction "set_origin_server" "www1.portal.com";
  } elseif [string match "*.mp3" $uri] {
   setaction "set_origin_server" "www2.portal.com";
  } else {
   setaction "set_origin_server" "www3.portal.com";
  }
  return;
}
```

Rejecting Referrer URLs

The example below shows how to reject the request if the requested URL is referred by Google.

```
proc pe_http_recv_request {} {
  if {[string match "google*" [getparam http.req.header Referer]]} {
   setaction "reject_request";
   return;
  }
  return;
}
```

Performing a 302 Redirect

The example below shows how to perform a 302 redirect.

```
proc pe_http_recv_request {} {
  if [string match "www.google.com" [getparam http.req.host]] {
   setaction "redirect" "http://www.yahoo.com/index.html";
   return;
  }
  return;
}
```

Tunneling Traffic

The example below shows how to tunnel traffic.

```
proc pe_http_recv_request {} {
  if [string match "10.*" [getparam socket.source_ip]] {
   setaction "no_cache_object";
   return;
  }
  return;
}
```

Appending a String to a Cache Name and Specifying Object Bandwidth Settings

This example shows how to append the string to the cache name and specify the bandwidth settings— maximum bandwidth, maximum bit rate for the video, and fast start size—for a given user connection.

```
proc pe_http_recv_request {} {

        set reqst [getparam http.req.header "User-Agent"];
        if [string match "iPhone" $reqst] {
                setaction cache_index_append "iphone";
                setaction transmit_rate "1Kbps" "1Kbps" "1MB";
                return;
        }

        return;
}
```

Using the URI to Fetch from Origin and Cache the Object in the New URI Name

This example shows how to rewrite the URL on the basis of the device from which the user request originated.

```
proc pe_http_recv_request {} {
        set reqst [getparam http.req.header "user-agent"];
        set uri [getparam http.req.uri];
        if [string match "*iPhone*" $reqst] {
                setaction url_rewrite iphone.cdn.com /2$uri;
                return;
        return;
}
```

Policy Examples Based on pe_om_recv_response

The examples in the following sub-sections illustrate policy scripts based on pe_om_recv_response.

Blocking User Requests Based on HTTP Headers

The example below shows how to block user requests based on HTTP headers.

```
proc pe_om_recv_response {} {
  if {[string match "*block*" [getparam http.req.header test_header]]} {
   setaction "reject_request";
   return;
  }
  if {[string match "Wget*" [getparam http.req.header User-Agent]]} {
   setaction "reject_request";
   return;
  }
  return;
}
```

Redirecting Users When the Origin Server Is Not Available

The example below shows how to redirect users when the origin server is not available.

```
proc pe_om_recv_response {} {
  set response_code [getparam om.res.status_code];
  if { [ string match "404" $response_code ] == 1 } {
   setaction "redirect" "http://www.mypartnerserver.com";
   return;
  }
  return;
}
```

Caching Objects

The examples below show how to cache objects. The first example uses cookies.

```
proc pe_om_recv_response {} {
  set cookie [getparam http.req.cookie test_cookie];
  if {[string compare "test_cookie=aaa" $cookie] == 0} {
   setaction "no_cache_object";
   return;
  }
  if [string match "*bbb*" $cookie] {
   setaction "cache_object";
   return;
```

```
  }
  return;
}
```

The second example uses a URI pattern.

```
proc pe_om_recv_response {} {
  set uri [getparam http.req.uri];
  if [string match "*.mpg" $uri] {
   setaction "cache_object";
   return;
  }
  if [regexp {vod[0-9].*mp4} $uri] {
   setaction "no_cache_object";
   return;
  }
  return;
}
```

Caching Content Marked “ no-cache” by the Content Provider

The example below shows how to cache content marked **no-cache** by the content provider.

```
proc pe_om_recv_response {} {
  if {[string match "*.mypartner.com" [getparam http.req.host]] && [string match
 "*no-cache*" [getparam om.res.header Cache-Control]]} {
   setaction "cache_object";
   return;
  }
  return;
}
```

Caching Content Marked “ private” by the Content Provider

The example below shows how to cache content marked **private** by the content provider.

```
proc pe_om_recv_response {} {
  if {[string match "*.youtube.com" [getparam http.req.host]] && [string match
"*private*" [getparam om.res.header Cache-Control]]} {
   setaction "cache_object";
   return;
  }
  return;
}
```

Overriding cache-control max-age

This example overrides the cache-control-max-age value set by the origin server. If no value is set, this command sets the value.

```
 proc pe_om_recv_response {} {

      if {[string match "*.\/dir1\/" [getparam http.req.uri]]} {
         set cc [getparam om.res.header Cache-Control]
          setaction "remove_header" “Age”;
          setaction "remove_header" “Expires”;
          setaction "modify_header" “Cache-Control: max-age=360”;
          return;
      }
```

```
      return;
}

proc pe_om_recv_response {} {

    if {[string match "*.flv" [getparam http.req.uri]]} {
       set cc [getparam om.res.header Cache-Control]
       setaction "remove_header" “Age”;
       setaction "remove_header" “Expires”;
       setaction "modify_header" “Cache-Control: max-age=360”;
       return;
    }
    return;
}
```

Do Not Cache Content Based on the Object's Size

The example below shows how to avoid caching content based on the object's size.

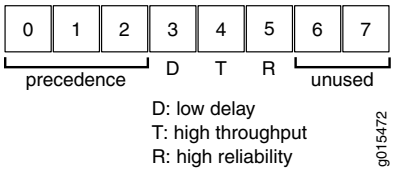
```
proc pe_om_recv_response {} {
  set len [getparam om.res.header Content-Length];
  if {$len <= 2024} {
   setaction "no_cache_object";
   return;
  }
  return;
}
```

Setting the Type of Service Bits

The examples below show how to set the type-of-service (ToS) bits.

The ToS bits are a six-bit field described in IETF RFC 791. The examples below use a prefix string of " 0b" to fill out the DSCP byte, for example: "0b110101". See Figure 18 on page 278 for an illustration of how the ToS bits are organized.

Figure 18: ToS Bits Description



The first example below shows how to use **pe_om_send_request** to set the IP ToS bits in a request to an origin server.

```
proc pe_om_send_request {} {
  if [string match "*.mpg" [getparam http.req.uri]] {
   setaction "set_ip_tos" 0b001111;
   return;
  }
  return;
}
```

The second example below shows how to use **pe_http_send_response** to set the IP ToS bits in a response to a client.

```
proc pe_http_send_response {} {
  if [string match "*.mpg" [getparam http.req.uri]] {
   setaction "set_ip_tos" 0b001111;
   return;
  }
  return;
}
```

The third example enables service differentiation based on user types.

```
proc pe_http_send_response {} {
   iif { [ string match "*user=premium*" [ getparam http.req.query ] ] } {
   setaction "set_ip_tos" 0b001111;
   return;
  }
  return;
}
```

The fourth example enables service differentiation based on request URI pattern.

```
proc pe_http_send_response {} {
  if {[string match "*.flv" [getparam http.req.uri]]} {
   setaction "set_ip_tos" 0b001111;
   return;
  }
  return;
}
```

The fifth example enables service differentiation based on the request's domain.

```
proc pe_http_send_response {} {
  if { [ string match "*.google.com" [ getparam http.req.host ] ] } {
   setaction "set_ip_tos" 0b001111;
   return;
  }
  return;
}
```

Using Geographical Data Tokens

The following examples illustrate GeoIP TCL scripts based on pe_http_recv_request.

- Changing the Origin Server on page 279
- Allowing Requests from a Specific Country on page 280
- Changing the URI for a Specific Country on page 280

Changing the Origin Server

Based on the client's location, different content could be served. In the example below, the URI part remains the same, and only the origin server is changed.

```
proc pe_http_recv_request {} {
 set ret [getparam “socket.geodata.status_code”];
 set resp [getparam “socket.geodata.country_code”];
 if {$ret == 200} {
  if [string match “US” $resp] {
   setaction set_origin_server “us.test.com”;
```

```
   return;
  }
 }
 return;
}
```

Allowing Requests from a Specific Country

Based on the client's location, only requests from a specific country are allowed. For other countries, the client is redirected to an error page.

```
proc pe_http_recv_request {} {
 set ret [getparam “socket.geodata.status_code”];
 set resp [getparam “socket.geodata.country_code”];
 if {$ret == 200} {
  if [string match “US” $resp] {
   return;
  }
 }
 setaction redirect “http://www.test.com/error.html”
 return;
}
```

Changing the URI for a Specific Country

Based on the client's location, different content could be served. In the example below, the URI is changed for a specific country.

```
proc pe_http_recv_request {} {
 set ret [getparam “socket.geodata.status_code”];
 set resp [getparam “socket.geodata.country_code”];
 set url [getparam “http.req.uri”];
 if {$ret == 200} {
  if [string match “US” $resp] {
   setaction url_rewrite “” /us$url;
   return;
  }
  if [string match “UK” $resp] {
   setaction url_rewrite “” /uk$url;
   return;
  }
 }
 return;
}
```

Related Documentation

- Setting the Type of Service Bits on page 278
- Using Geographical Data Tokens on page 279
- Policy Engine Overview on page 245

CHAPTER 11

Configuring Media Flow Controller (Web Interface)

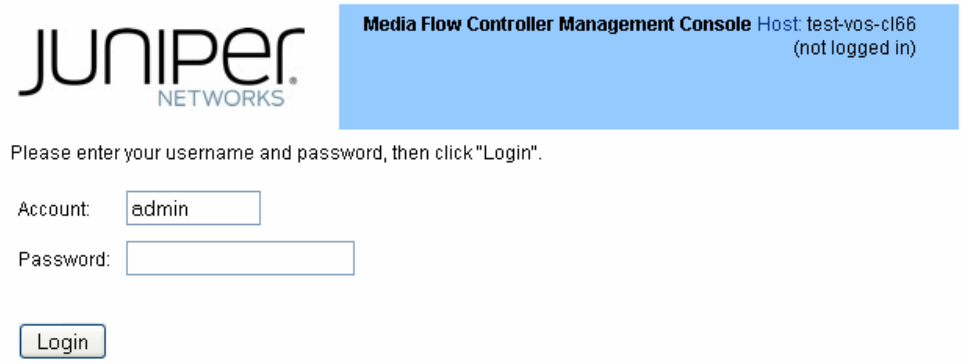
About the Media Flow Controller Web Interface

About the Media Flow Controller Web Interface

Before you configure Media Flow Controller for the first time, see "Before You Configure Media Flow Controller" on page 20.

The Media Flow Controller Web interface, also referred to as the Management Console, provides a subset of the command-line interface management functions; however, you can make many configurations with the Web interface. See Figure 19 on page 282.

Figure 19: Media Flow Controller Login Page



To configure Media Flow Controller using the Media Flow Controller Management Console Web Interface, the Web UI and HTTP access to it must be enabled on the management interface, Eth0. These settings are enabled by default; however, if you need to re-enable them, you must use the CLI **web** commands.

Connecting and Logging In

You can connect to the Web interface using the IP address of your Media Flow Controller in a browser on port 8080. The Media Flow Controller opens to a login page.

Each user account has at least one privilege level that determines what actions they can take in the Web interface Management Console:

- Administrator (**Monitoring Summaryadmin**)—Full privileges. Can enter **Enable** mode and **Config** mode. By default, can log in as **admin** without a password.
- Monitor (**monitor**)—Can view all configurations but cannot change any configurations. Can view Dashboard and Reports, but cannot view Logs. By default, can log in as **monitor** without a password.
- Unprivileged (**unpriv**)—Can view all configurations but cannot change any configurations. Can view Dashboard and Reports, but cannot view Logs. Cannot log in as **unpriv**.

Related Documentation

- Media Flow Controller Deployments Overview on page 447
- Media Flow Controller Environment on page 4
- Media Flow Controller Minimum System Requirements on page 4

- Understanding Media Flow Controller on page 5

Logging In to Media Flow Controller for the First Time (Web Interface)

Before you log in to Media Flow Controller for the first time, see "Before You Configure Media Flow Controller" on page 20.

To log in to the Media Flow Controller Web interface for the first time, you need the IP address assigned to the Eth0 management interface.

1. Open a browser and enter the Media Flow Controller management IP address including the management port, 8080. For example:

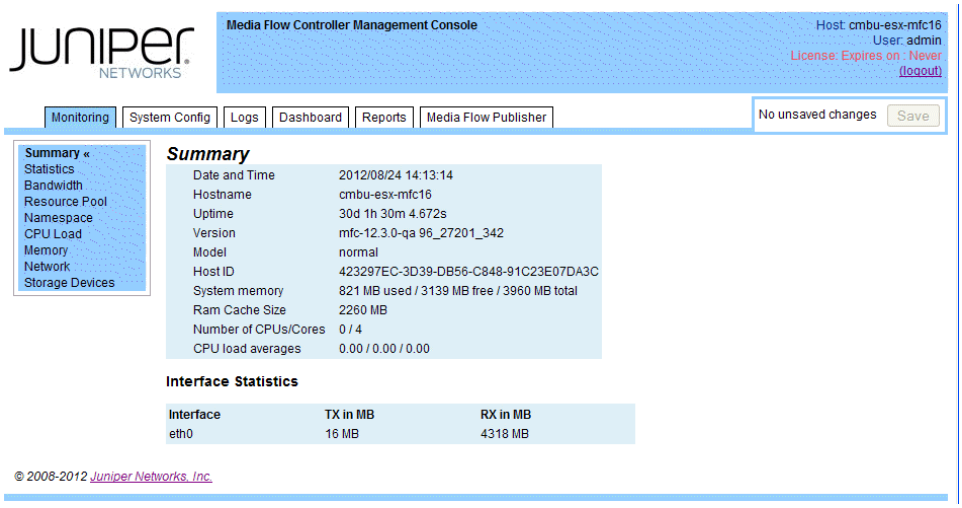
   http://192.168.1.100:8080

2. Log in with these default credentials (there is no default password).
   User: **admin**

   The **Monitoring Summary** page appears.

   See Figure 20 on page 283.

Figure 20: Monitoring Summary



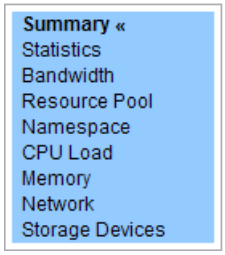
Related Documentation

- Monitoring Media Flow Controller Statistics (Web Interface) on page 284
- Configuring Faults and Logging (Web Interface) on page 309
- System Configuration Overview (Web Interface) on page 287
- About the Media Flow Controller Web Interface on page 281
- Before You Configure Media Flow Controller on page 20

Monitoring Media Flow Controller Statistics (Web Interface)

The **Monitoring** tab gives you quick access to statistics and information about the current system, including bandwidth usage, namespace usage, CPU load, and more. Figure 21 on page 284 shows the **Monitoring** menu.

Figure 21: Monitoring tab left navigation menu



Viewing Media Flow Controller Summary

Purpose View statistics and graphs of the managed Media Flow Controllers; and this information:

- Summary
  - Date and Time
  - Hostname
  - Uptime
  - Version
  - Partitioning
  - Hardware
  - Host ID (system serial string on the motherboard: your licenses are tied to the Host ID)
  - System Memory (used, free, and total)
  - RAM Cache Size

  - Number of CPUs/Cores
  - CPU load averages.
- **Interface Statistics**—The current TX (transmissions out) and RX (transmissions received) system statistics.

Action After you log in to Media Flow Controller, select the **Monitoring** tab.
The **Summary** page appears.

Viewing Media Flow Controller Statistics

Purpose View counters for:

- Cache Hierarchy
- HTTP Origin
- Virtual Player
- Connections
- HTTP Delivery
- HTTP Tunnel
- Real-Time Streaming Delivery

View statistics **Current Value**, **Alarm Threshold**, and **Clear Threshold** for:

- Current Bandwidth (MB/Sec)
- Average Cache Bandwidth (MB/Sec)
- Average Disk Bandwidth (MB/Sec)
- Average Origin Bandwidth (MB/Sec)
- Average Connection Rate (per Sec)
- Average HTTP Transaction Rate (per Sec)
- Current Cache Bandwidth (MB/Sec)
- Current Disk Bandwidth (MB/Sec)
- Current Origin Bandwidth (MB/Sec)
- Free Filesystem (%)
- Average CPU Utilization (%)
- Paging

Action
- From the left navigation pane in the **Monitoring** tab, click **Statistics**.
  The **Statistics Summary** page appears.
- Click **Reset Counter** at the top of the page.
  All statistics counters are restarted.

Viewing Media Flow Controller Bandwidth Usage

Purpose Check **Cache Bandwidth**, **Disk Bandwidth**, and **Origin Bandwidth** usage for the last hour.

Action
- From the left navigation pane in the **Monitoring** tab, click **Bandwidth**.
  The **Bandwidth Usage (Last Hour)** page appears.
- Click **Pause** and **Resume** buttons to stop/start graph charting.

Viewing Media Flow Controller Resource Pool Stats

Purpose List the currently configured resource pools and the current settings for each: **Resource Pool Name**, **Associated Namespaces**, **Resource Type**, **Maximum**, and **Current**.

Action
- From the left navigation pane in the **Monitoring** tab, click **Resource Pool**.
  The **Resource Pool Stats** page appears.

Viewing Media Flow Controller Namespace Counters

Purpose List the currently configured namespaces and the current number of GET requests for each: **Number of Requests**, **Resource Pool**, **HTTP: Current Sessions**, **HTTP: Bandwidth (Mbps)**, **Cache Hit Ratio: Based on Request**, and **Cache Hit Ratio: Based on Bytes**.

Action
- From the left navigation pane in the **Monitoring** tab, click **Namespace**.
  The **Namespace Counters** page appears.

Viewing Media Flow Controller CPU Load

Purpose View different graphs of the CPU load in the last hour.

Action
- From the left navigation pane in the **Monitoring** tab, click **CPU Load**.
  The **CPU Load (Last Hour)** page appears.
- Select from the drop-down menu to view **Aggregated** (default), **Per CPU**, or **Per CPU Stacked** graph.
- Use the **Pause** and **Resume** buttons to stop/start graph charting, use **Clear Data** to start new graphing.

Viewing Memory Utilization

Purpose Display a last-day graph of **Memory Utilization** plus a pie chart of **Current Memory Statistics** including statistics for Physical and Swap memory (Total, Used, and Free; and Buffered and Cached for Physical).

Action
- From the left navigation pane in the **Monitoring** tab, click **Memory**.
  The **Memory Utilization (Last Day)** page appears.
- Use the **Pause** and **Resume** buttons to stop and start graph charting.

Viewing Network Usage

Purpose Display a last-hour graph of **Network Usage**, including RX and TX information on all data ports. Current statistics include:

- Bytes, packets, discards, errors, and overruns for RX and TX
- RX mcast packets
- RX frame
- TX carrier
- TX collisions

Action

- From the left navigation pane in the **Monitoring** tab, click **Network**.

  The **Network Usage (Last Hour)** page appears.
- Use the **Pause** and **Resume** buttons to stop or start graph charting.

Viewing Storage Device Usage

Purpose Display **Storage Device Usage** sda usage (since boot): **Total data read** and **Total data written**.

Action

- From the left navigation pane in the **Monitoring** tab, click **Storage Devices**.

  The **Storage Device Usage** page appears.

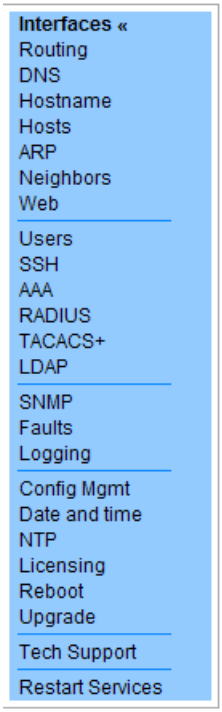
Related Documentation

- Logging In to Media Flow Controller for the First Time (Web Interface) on page 283
- System Configuration Overview (Web Interface) on page 287
- About the Media Flow Controller Web Interface on page 281

System Configuration Overview (Web Interface)

You can configure many system settings for the Juniper Networks Media Flow Controller by using the **System Config** tab. Figure 22 on page 288 shows the **System Config** menu.

Figure 22: System Config Tab left navigation menu



To configure system settings:

- Configure the interfaces.
  See "Configuring Interfaces, Default Gateway, Static Routes, DNS and Domain Names, Hostname, and Banners (Web Interface)" on page 289.
- Configure routing parameters.
  See "Configuring Static Hosts and ARP (Web Interface)" on page 295.
- Configure system date, time, and NTP servers.
  See "Configuring Date, Time, and NTP (Web Interface)" on page 298.
- Configure authentication and authorization options, and users.
  See "Configuring Users and AAA (Web Interface)" on page 305.
- Configure fault notifications, and logging options.
  See "Configuring Faults and Logging (Web Interface)" on page 309.

Related Documentation

- About the Media Flow Controller Web Interface on page 281
- Logging In to Media Flow Controller for the First Time (Web Interface) on page 283
- Managing Configuration Files (Web Interface) on page 316

Configuring Interfaces, Default Gateway, Static Routes, DNS and Domain Names, Hostname, and Banners (Web Interface)

Before you configure Media Flow Controller interfaces, see "Before You Configure Media Flow Controller" on page 20.

TIP: You may want to change the Web default logout time (900 = 15 minutes); to do this:

- Click the **System Config** tab.
  The **Network Interfaces** page appears.

Configuring Interfaces (Web Interface)

We recommend using Eth0 for management, eth1 for origin fetch, and the other interfaces for traffic. See "Example: Media Flow Controller Interface Configuration" on page 39 for details. You can also configure the DHCP primary interface and add interface aliases on this page.

To view and set network interfaces:

- From the left navigation pane in the **System Config** tab, select **Interfaces**.
  The **Network Interfaces** page appears.

To restart mod-delivery:

- Click **Restart Services**.

Viewing the Eth0 State

Displays the state of the Eth0 interface.
To view the **eth0 state** area:

- **Status**
  - **Admin Up**: The interface is enabled.

  - **Link Up**: The interface has a current connection.
- **IP address**
- **Netmask**
- **Type**
- **Speed**
- **Duplex**
- **MTU**
- **HWaddr**: Displays the hardware address.

Configuring Eth0

The Eth0 interface is used to manage Media Flow Controller.

CAUTION: For VXA Series Media Flow Engine appliances, do not ever change the Ethernet name mappings; all interface assignments are handled automatically during manufacturing.

To configure the Eth0 interface on the **Network Interfaces** page:

1. Enter information in the text box:
   - **Enabled**—Select to enable the interface for activity, default; or deselect the check box to disable the interface, respectively.
   - **Obtain IP Address Automatically (DHCP)**—Allow DHCP to assign the IP address for this Media Flow Controller. OR
   - **Specify IP Address Manually**—Enter the **IP address** and **Netmask** you want for this Media Flow Controller.
   - **Speed** and **Duplex**—Select **Auto** (default) for the Speed and Duplex to be set automatically based on hardware. Or you can set these options to alternate values in the drop-down menu. We highly recommend that **Speed** and **Duplex** not be changed from the auto-configured defaults.
   - **MTU**—The Maximum Transmission Unit (MTU) sets the largest number of bytes a frame can carry. Default is 1500.
   - **Comment**—A description for the interface.
2. Click **Apply** to immediately apply changes; click **Cancel** to revert to the existing configuration.
3. Click **Save** at the top of the page to make changes persistent.

Setting the DHCP Primary Interface

Set the Dynamic Host Configuration Protocol (DHCP) primary interface and verify the current primary DHCP interface.

DHCP allows new network devices to be automatically supplied with an IP address and other information, depending on the setup of the DHCP server. Media Flow Controller has no primary DHCP interface by default. Setting a primary interface ensures that DHCP messages arrive only on that interface.

To set the DHCP Primary interface:

1. Select a **Configured primary** interface from the drop-down list box box.
2. Click **Apply** to immediately apply changes; click **Cancel** to revert to existing configuration.
3. Click **Save** at the top of the page to make changes persistent.

Adding New Interface Alias

An interface alias lets you assign multiple IP addresses to the same interface.

To add a new interface alias:

1. Select an existing interface from the drop-down list box.
2. Enter the following information in the text box:
   - **Alias index**—A name for the interface alias. This creates a pseudo interface; its address appears in the output of **show interface** under the primary interface data.
   - **IP address**
   - **Netmask**
3. Click **Apply** to immediately apply changes; click **Cancel** to revert to existing configuration. If you click **Apply**, **state** and **configuration** areas for the new alias are displayed. In the configuration area, you can change disable the interface (interfaces are enabled by default), change the **IP address** and **Netmask**, and add a **Comment**.
4. Click **Save** at the top of the page to make changes persistent.

Setting the Default Gateway and Static Routes (Web Interface)

Set IP routing parameters, including **Default Gateway** and **Static Routes**. The default gateway provides an entry and exit point between the internal network and all external networks; requests with destinations outside of the internal network's routing table are sent to the default gateway for routing.

Static routes, configured paths to a subnet, are used to route data from one subnet to another.

To set network IP routing parameters:

- From the left navigation pane in the System Config tab, select **Routing**. The **IP Routing** page appears.

Setting the Default Gateway

Set the default gateway as the main access point to external networks, including the Internet.

To set the default gateway:

1. Enter an IP address in the Default gateway text box and click **Set Default Gateway** to immediately apply changes.
2. Click **Save** at the top of the page to make them persistent across reboots.

Viewing Static and Dynamic Routes

A static route is a hard coded (manually defined) path that specifies the route to a certain subnet using a certain path.

View all configured static and dynamic routes for IPv4 and IPv6:

- **Destination**—The subnet/path for this static route.
- **Mask or mask length**—The netmask for this route.
- **Gateway**—The configured gateway (path to the Internet) for this static route.
- **Interface**—The port configured for this static route.
- **Source**—The source for this static route.
- **Active**—Whether or not this route is being used currently.
- **Static**—Whether or not this route is static (hard coded).

To remove a static route:

- Select a route and click **Removed Selected** to immediately apply changes; click **Save** at the top of the page to make them persistent across reboots.

Adding a Static Route

Static routes set a path in the routing table for a particular destination.

To set a static route:

1. Enter the following information in the text boxes:
   - **Destination**—A network prefix for where you want a static route to.
   - **Netmask**—The netmask for the configured network prefix.

- **Gateway**—IP address of the configured default gateway.
- **Interface**—Select a configured interface from the drop-down list box.

2. Click **Add Route** to immediately apply changes.
3. Click **Save** at the top of the page to make them persistent across reboots.

Configuring DNS and Domain Names (Web Interface)

View, add, modify, or remove **Static and Dynamic Name Servers** and **Domain Names** from your Domain Name System (DNS).

To configure DNS and domain names:

- From the left navigation pane in the **System Config** tab, select **DNS**.
  The **DNS** page appears.

Viewing Static and Dynamic Name Servers

View all configured static and dynamic name servers:

- **IP Address**—IP address configured name server.
- **Active**—Whether or not this name server is being used currently.
- **Source**—"Configured" means it was manually added; "Dynamic" means it came from a name server.

Adding or Modifying Name Servers

To add or modify name servers from the **DNS** page:

1. You can set up to three dynamic name servers; for each option, enter an IP address.
   - **Primary DNS IP address**—This name server is tried first.
   - **Secondary DNS IP address**—This name server is tried second.
   - **Tertiary DNS IP address**—This name server is tried last.
2. Click **Apply** to immediately apply changes; click **Cancel** to revert to existing configuration.
3. Click **Save** at the top of the page to make changes persistent.

Viewing Static and Dynamic Domain Names

View all configured static and dynamic domain names:

- **Domain**—The configured name for that domain.
- **Active**—Whether or not this domain name is being used currently.
- **Source**—"Configured" means it was manually added; "Dynamic" means it came from a name server.

Deleting Configured Domain Names

To delete a domain name from the **DNS** page:

1. Select a configured domain name from the list and click **Remove Selected**.
2. Click **Save** at the top of the page to make changes persistent.

Adding a New Domain Name

To add a domain name from the **DNS** page:

1. Enter a **Domain Name** and click **Add Domain Name**; it can be removed using the list of **Configured Domain Names** described above.
2. Click **Save** at the top of the page to make changes persistent.

Setting Hostnames and Banners (Web Interface)

View or change the **System Hostname** and the **DHCP Hostname** (Dynamic Host Configuration Protocol), and set Banners. You can set a **MOTD** (message of the day), a **Login Remote**, and a **Login Local** banner.

To configure the system hostname, the DHCP hostname and options, and banners:

- From the left navigation pane in the **System Config** tab, select **Hostname**. The **Hostname and Banners** page appears.

- Configuring a System Hostname on page 294
- Specifying a DHCP Hostname on page 294
- Setting Banners on page 295

Configuring a System Hostname

To configure a hostname for the system on the **Hostnames and Banners** page:

1. Enter a name in the **Host Name** text box and click **Apply** to immediately apply changes; click **Cancel** to revert to existing configuration.
2. Click **Save** at the top of the page to make changes persistent.

Specifying a DHCP Hostname

You can configure Dynamic Host Configuration Protocol (DHCP) settings. An interface is only eligible to be the DHCP primary if it is "admin up," has DHCP enabled, and has a DHCP lease. Ineligibility does not prevent an interface from being configured as the DHCP primary interface, but prevents it from actually being used as such. If the configured

primary interface is eligible, it is chosen as the acting primary; otherwise, one of the eligible interfaces is chosen (the first, in alphabetical order).

To configure a hostname for the Dynamic Host Configuration Protocol on the **Hostnames and Banners** page:

1. Select or leave deselected the **Send hostname with DHCP client request** check box. By default, no hostname is sent during DHCP negotiation. The server can use and honor the hostname supplied by the client. By default, the client sends the system's hostname. This can be overridden by entering an **Alternate hostname for DHCP**. This is a global configuration command that affects all DHCP interfaces. Changing this setting forces a renewal of DHCP on all interfaces.
2. Select a DHCP hostname to be sent with the DHCP client request (if selected):
   - **Use system hostname**—Default.
   - **Alternate hostname for the DHCP**— Only applies when **Send hostname with DHCP client request** is selected. The hostname can be unqualified or fully qualified. This is a global configuration command that affects all DHCP interfaces. Changing this setting forces a renewal of DHCP on all interfaces.
3. Click **Save** at the top of the page to make changes persistent.

Setting Banners

You can set a **MOTD** (message of the day), a **Login Remote**, and a **Login Local** banner.

To set a banner for the system on the **Hostnames and Banners** page:

1. To add a **MOTD**, **Login Remote**, or **Login Local banner**, enter the text in the specified box. Click **Apply** to immediately apply changes; click **Cancel** to revert to the existing configuration.
2. Click **Save** at the top of the page to make changes persistent.

Related Documentation

- Logging In to Media Flow Controller for the First Time (Web Interface) on page 283
- System Configuration Overview (Web Interface) on page 287
- Administering Media Flow Controller Overview on page 315
- Monitoring Media Flow Controller Statistics (Web Interface) on page 284

Configuring Static Hosts and ARP (Web Interface)

Before you configure Media Flow Controller static hosts and ARP, see "Before You Configure Media Flow Controller" on page 20.

- Configuring Static Hosts on page 296
- Configuring ARP (Web Interface) on page 296

Configuring Static Hosts

A static host is not subject to IP address changes using Dynamic Name Servers (DNSs). To set static hosts:

- From the left navigation pane in the **System Config** tab, select **Hosts**. The **Static Hosts** page appears.

- Viewing Static Host Entries on page 296
- Adding a New Host on page 296

Viewing Static Host Entries

View configured static host entries, including **IP address** and **Hostname**.

To delete a static host entry on the **Static Hosts** page:

1. Click **Remove Selected** to delete any static host entries you have created.
2. Click **Save** at the top of the page to make changes persistent.

Adding a New Host

You must know the **IP address** and **Hostname** to enter a static host mapping.

To add a new host on the **Static Hosts** page:

1. Enter an **IP address** and **Hostname**. Click **Add Entry** to immediately apply changes.
2. Click **Save** at the top of the page to make changes persistent.

Configuring ARP (Web Interface)

To manage Address Resolution Protocol (ARP) entries on the **Static Hosts** page:

- From the left navigation pane in the **System Config** tab, select **ARP**. The **Address Resolution** page appears.

- Viewing Static and Dynamic ARP Entries on page 296
- Adding a Static Entry on page 297
- Clearing the Dynamic ARP Cache on page 297

Viewing Static and Dynamic ARP Entries

View and remove **Static and Dynamic ARP Entries**.

To view and remove static and dynamic ARP entry status from the **Address Resolution** page:

1. View the following information:
   - **IP address**—The configured IP address for this entry.
   - **MAC address**—The physical address of this entry.

- **Interface**—The port configured for this entry.
- **Active**—Whether or not this entry is being used currently.
- **Static**—Whether or not this entry comes from DNS.

2. Click **Remove Selected** to delete an entry.
3. Click **Save** at the top of the page to make changes persistent.

Adding a Static Entry

To add a static ARP entry from the **Address Resolution** page:

1. Enter the **IP address** and **MAC address** of the system you want for ARP.
2. Click **Add Entry** to immediately apply changes; click **Cancel** to revert to existing configuration.
3. Click **Save** at the top of the page to make changes persistent.

Clearing the Dynamic ARP Cache

Click **Clear** to empty the ARP cache.

Related Documentation

- Logging In to Media Flow Controller for the First Time (Web Interface) on page 283
- System Configuration Overview (Web Interface) on page 287
- Administering Media Flow Controller Overview on page 315
- Monitoring Media Flow Controller Statistics (Web Interface) on page 284

Configuring IPv6 NDP (Web Interface)

You can configure, view, and clear IPv6 Neighbor Discovery Protocol (NDP) entries.

Viewing Static and Dynamic NDP Entries

- From the left navigation pane in the **System Config** tab, select **Neighbors**. The **IPv6 Neighbor Discovery Protocol** page appears.

View information on configured static and dynamic NDP entries:

- **IPv6 Address**
- **MAC Address**
- **Interface**
- **Age (secs)**

- **NUD State**: Neighbor Unreachability Detection state
- **Active**
- **Static**

To delete an NDP entry:

1. On the IPv6 Neighbor Discovery Protocol page, click **Remove Selected** to delete any NDP entries you have created.
2. Click **Save** at the top of the page to make changes persistent.

Adding a Static Entry

You must know the **IP address** and **MAC Address** to enter a static entry.

To add a new static entry:

1. On the **IPv6 Neighbor Discovery Protocol** page, enter an **IP address** and **MAC Address**, and select an **Interface**. Click **Add Entry** to immediately apply changes.
2. Click **Save** at the top of the page to make changes persistent.

Clearing the Dynamic ARP Cache

Click **Clear** to empty the IPv6 Dynamic NDP cache.

Related Documentation

- Logging In to Media Flow Controller for the First Time (Web Interface) on page 283
- System Configuration Overview (Web Interface) on page 287
- Administering Media Flow Controller Overview on page 315
- Monitoring Media Flow Controller Statistics (Web Interface) on page 284

Configuring Date, Time, and NTP (Web Interface)

Before you configure Media Flow Controller date, time, and NTP, see "Before You Configure Media Flow Controller" on page 20.

- Configuring the System Date and Time (Web Interface) on page 298
- Configuring NTP (Web Interface) on page 299

Configuring the System Date and Time (Web Interface)

Proper time configuration is required for correct caching (validating content, and so forth).

To configure the system date and time:

- From the left navigation pane in the **System Config** tab, select **Date and time**. The **Date and Time** page appears.

To configure the system date, time, and timezone:

1. Enter the following information:
   - Date
   - Time
2. Select a **Time Zone** from the drop-down list box.
3. (Optional) Click the enabled link to jump to the **NTP** page.
4. Click **Apply** to set the date, time, and time zone; click **Cancel** to revert to existing configuration.
5. Click **Save** at the top of the page to make changes persistent across reboots.

Configuring NTP (Web Interface)

To configure Network Time Protocol (NTP) servers:

- From the left navigation pane in the **System Config** tab, select **NTP**.
  The **NTP** page appears.

- Setting Up NTP on page 299
- Managing NTP Servers on page 299
- Adding a New NTP Server on page 300

Setting Up NTP

To enable or disable NTP time synchronization:

1. Select the check box to enable NTP; deselect it to disable NTP.
2. Click **Apply** to complete enabling NTP, click **Cancel** to revert to existing configuration.
3. Click **Save** at the top of the page to make changes persistent across reboots.

Managing NTP Servers

To view and manage NTP servers.

1. For each configured NTP server, view the following information:
   - **Server**—IP address of the NTP server.
   - **Status**—Whether or not the server is currently in use.
   - **Stratum**—The hierarchical system this NTP server uses.
   - **Offset (ms)**—Whether or not an offset (a degree in milliseconds of difference) of the server's time is configured.
   - **Reference Clock**—How the NTP server is finding its reference (base) clock; INIT means this is configured in the /etc/init.d directory.
   - **Poll Interval (sec.)**—How often the server exchanges information with its configured reference clocks.

- **Last response (sec.)**—When the last response from the reference clock was received.
- **NTP version**—The configured version of NTP.

2. Select a server and click **Remove Selected Server**, **Enable Server**, or **Disable Server**.
3. Click **Save** at the top of the page to make changes persistent across reboots.

Adding a New NTP Server

Add additional NTP servers for redundancy.

To add a new NTP server:

1. Enter a **Server IP** address. Select a **Version** from the drop-down list box and select **Yes** or **No** from the **Enabled** drop-down list box to enable or disable the NTP server. Click **Add NTP Server**.
2. Click **Save** at the top of the page to make changes persistent across reboots.

Related Documentation

- Logging In to Media Flow Controller for the First Time (Web Interface) on page 283
- System Configuration Overview (Web Interface) on page 287
- Administering Media Flow Controller Overview on page 315
- Monitoring Media Flow Controller Statistics (Web Interface) on page 284

Configuring RADIUS, TACACS+, LDAP, and SSH (Web Interface)

Before you configure Media Flow Controller for the first time, see "Before You Configure Media Flow Controller" on page 20.

NOTE: **Media Flow Controller supports only authentication service for the TACACS+ protocol.**

- Configuring RADIUS (Web Interface) on page 300
- Configuring TACACS+ (Web Interface) on page 302
- Configuring LDAP (Web Interface) on page 303
- Configuring SSH (Web Interface) on page 305

Configuring RADIUS (Web Interface)

To configure RADIUS:

- From the left navigation pane in the **System Config** tab, select **RADIUS**.
  The **RADIUS** page appears.

- Configuring Default RADIUS Settings on page 301
- Managing RADIUS Servers on page 301
- Adding a New RADIUS Server on page 301

Configuring Default RADIUS Settings

To configure default RADIUS settings:

1. Enter the following information in the text boxes:
   - **Key**—A shared secret text string. If no **key** is set, the user is prompted for the key.
   - **Timeout**—Timeout for retransmitting a request to any RADIUS server. Range is **1** through **60**; the default is **3**.
   - **Retransmit**—The number of times the client attempts to authenticate with any RADIUS server. Range is **0** through **5**; the default is **1**.
2. Click **Apply** to immediately apply changes; click **Cancel** to reset previous values.
3. Click **Save** at the top of the page to make changes persistent across reboots.

Managing RADIUS Servers

You can manage RADIUS servers in the server list: **Remove**, **Enable**, or **Disable RADIUS Servers**.

To manage Radius servers:

1. View the following information on configured RADIUS servers:
   - **Server**—The configured IP address for this RADIUS server.
   - **Auth-Port**—The configured port for authentication requests to this server.
   - **Key**—The configured shared secret text string. If empty, the user is prompted for the key.
   - **Timeout**—The configured timeout for retransmitting a request.
   - **Retransmit**—The configured number of times a client may attempt to authenticate.
   - **Enabled**—Whether or not this RADIUS server is enabled. Disabling a server makes it inactive but does not delete it from the system.
2. Select a server and select **Remove Selected Server**, **Enable Server**, or **Disable Server**.
3. Click **Save** at the top of the page to make changes persistent across reboots.

Adding a New RADIUS Server

To add and enable a new RADIUS server:

1. Select **Enabled**—The server must be enabled to do authentication.
2. Enter the following information in the text boxes:
   - **Server IP**—IP address for the server.
   - **Auth Port**—The port for authentication requests; the default is **1812**. You can use the same IP address in more than one **host** as long as the **auth-port** is different.
3. (Optional) Enter overrides for the **Key**, **Timeout**, and **Retransmit** values for this RADIUS server from the default RADIUS settings you made above.
4. Click **Add RADIUS Server** to complete operation.
5. Click **Save** at the top of the page to make changes persistent across reboots.

Configuring TACACS+ (Web Interface)

Configure TACACS+ authentication options.
To configure TACACS+:

- From the left navigation pane in the **System Config** tab, select **TACACS+**.
  The **TACACS+** page appears.

- Configuring Default TACACS+ Settings on page 302
- Managing TACACS+ Servers on page 302
- Adding a New TACACS+ Server on page 303

Configuring Default TACACS+ Settings

To configure **Default TACACS+ settings**:

1. Enter the following information in the text boxes:
   - **Key**—A shared secret text string. If no **key** is set, the user is prompted for the key.
   - **Timeout**—Timeout for retransmitting a request. Range is **1** through **60**; the default is **3**.
   - **Retransmit**—The number of times the client attempts to authenticate with any TACACS+ server. Range is **0** through **5**; the default is **1**.
2. Click **Apply** to immediately apply changes; click **Cancel** to reset previous values.
3. Click **Save** at the top of the page to make changes persistent across reboots.

Managing TACACS+ Servers

To manage TACACS+ servers, use the following in the server list: **Remove**, **Enable** or **Disable TACACS+ Servers**.

1. View the following information on configured TACACS+ servers:
   - **Server**—The configured IP address for this TACACS+ server.
   - **Auth-Port**—The configured port for authentication requests to this server.

- **Auth-Type**—The configured type of authentication this TACACS+ server will use.
- **Key**— The configured shared secret string. If empty, the user is prompted for the key.
- **Timeout**—The configured timeout for retransmitting a request.
- **Retransmit**—The configured number of times a client may attempt to authenticate.
- **Enabled**—Whether or not this TACACS+ server is enabled. Disabling a server makes it inactive but does not delete it from the system.

2. Select a server and select **Remove Selected Server**, **Enable Server**, or **Disable Server**.
3. Click **Save** at the top of the page to make changes persistent across reboots.

Adding a New TACACS+ Server

To add and enable a new TACACS+ server:

1. Select **Enabled**—The server must be enabled to do authentication.
2. Enter the following information in the text boxes:
   - **Server IP**—IP address for the server.
   - **Auth Port**—The port for authentication requests; the default is **49**. You can use the same IP address in more than one host as long as the **auth-port** is different.
   - **Auth Type**—Which type of authentication this TACACS+ server will use; both authentication types transmit the username and password in un-encrypted text and are acceptable when passwords are stored in an external database. Select either:
     - **ascii**—American Standard Code for Information Interchange.
     - **pap**—Password authentication protocol (default).
3. (Optional) Enter overrides for **Key**, **Timeout**, and **Retransmit**, values for this TACACS+ server from the default TACACS+ settings you made above.
4. Click **Add TACACS+ Server** to complete the operation.
5. Click **Save** at the top of the page to make changes persistent across reboots.

Configuring LDAP (Web Interface)

To configure LDAP options:

- From the left navigation pane in the **System Config** tab, select **LDAP**.
  The **LDAP** page appears.

- Configuring Global LDAP Settings on page 304
- Removing LDAP Servers on page 304
- Adding a New LDAP Server on page 304

Configuring Global LDAP Settings

To configure **Global LDAP Settings**:

1. Enter the following information in the text boxes:
   - **User base DN**
   - **User search scope**—Choose **Subtree**, **One Level**, or **No Search**.
   - **Bind DN**
   - **Bind password**
   - **Group base DN**
   - **Group attribute**
   - **LDAP Version**—Choose **2** or **3**.
   - **Referrals**
   - **Search port**
   - **Search timeout**
   - **Bind timeout**
   - **SSL mode**—Choose **SSL**, **TLS**, or **none**.
   - **Server SSL port**
   - **SSL cert verify**
2. Click **Apply** to immediately apply changes; click **Cancel** to reset to previous values.
3. Click **Save** at the top of the page to make changes persistent across reboots.

Removing LDAP Servers

To remove an LDAP server:

1. View the list of **LDAP Servers**.
2. Select a server and **Remove Selected Server**.
3. Click **Save** at the top of the page to make changes persistent across reboots.

Adding a New LDAP Server

To add a new LDAP server on the **Add New LDAP Server** area:

1. Enter the **Server IP** address.
2. Click **Add LDAP Server** to complete operation.
3. Click **Save** at the top of the page to make changes persistent across reboots.

Configuring SSH (Web Interface)

Configure SSH transmissions. SSH is a protocol used to secure connections through an encryption known to the server and the client. The encryption operates using Host Keys, a secret string configured on the server and communicated to an authenticated client. See **ssh** in the *Media Flow CLI Controller Command Reference* for CLI details.

To configure SSH:

- From the left navigation pane in the **System Config** tab, select **SSH**.

- Viewing SSH Servers on page 305
- Generating New Host Keys on page 305

Viewing SSH Servers

View the following information on the current **SSH Server Host Keys:**

- **Key Type**—Either RSA1 (inventors initials) or DSA2 (Digital Signature Algorithm, 2).
- **Finger Print**—A human-readable string so you can check the key manually.

Generating New Host Keys

To generate a new identity (private and public keys) for the logged-in user, click **Generate**.

When the keys are generated, the private key is written to the logged-in user's .ssh directory in an appropriately named file (for example, id_dsa). This identity can be used when the user connects from the system to another host with **slogin**.

Related Documentation

- Logging In to Media Flow Controller for the First Time (Web Interface) on page 283
- System Configuration Overview (Web Interface) on page 287
- Administering Media Flow Controller Overview on page 315
- Monitoring Media Flow Controller Statistics (Web Interface) on page 284

Configuring Users and AAA (Web Interface)

Before you configure Media Flow Controller for the first time, see "Before You Configure Media Flow Controller" on page 20.

Before configuring Authentication, Authorization, and Accounting (AAA) , configure your authentication methods; RADIUS and TACACS+ are supported authentication methods.

- Configuring Users (Web Interface) on page 306
- Configuring AAA (Web Interface) on page 308

Configuring Users (Web Interface)

Configuring Users (Web Interface)

View, remove, enable, disable, and add new users; plus change existing user passwords.

To manage user accounts:

- From the left navigation pane in the **System Config** tab, select **Users**.
  The **Users** page appears.

Viewing Active Users

View configured users information.

To view **Active User** information:

- **Username**—Used for logins.
- **Full Name**—For the user, as configured.
- **Line**—How the user is connected: SSH or Web.
- **Host**—What system this user is configured on.
- **Idle (seconds)**—Time since the last command execution of this user.

Managing User Accounts

Manage **User Accounts** information.

For each configured user account:

1. View user account information:
   - **User**—Used for logins.
   - **Full Name**—For the user, as configured.
   - **Capability**—The privilege level assigned this user. There are three pre-defined capabilities:
     - **admin**—Full privileges (default); in **Enable** mode all **EXEC** commands are available.
     - **monitor**—Privileges for reading configuration data (not logs) and performing all actions, but not for changing any configuration.
     - **unpriv**—Unprivileged.

- **Account Status**—Password and login information.
- **Enabled**—Whether or not this user account is enabled. User accounts are enabled by default. Disabling an account makes it inactive (logins are disallowed) but does not delete it from the system.

2. Click **Edit** at the end of the row to change the information for a user, including passwords.
3. Select a user account and select **Remove Selected User**, **Enable User**, or **Disable User**.
4. Click **Save** at the top of the page to make changes persistent across reboots.

Adding a New User

To add a new user:

- Click **Add New User**.
  A new screen displays: **Add New User**.

1. Enter the following information in the text boxes:
   - **User**—Login name.
   - **Full Name**—Displays in the User Accounts list.
   - **Enabled**—Choose **Yes** or **No**.
   - **Capability**—Choose one of three pre-defined capabilities:
     - **admin**—Full privileges (default); in **Enable** mode all **EXEC** commands are available.
     - **monitor**—Privileges for reading configuration data (not logs) and performing all actions, but not for changing any configuration.
     - **unpriv**—Unprivileged.
   - **Account status**—Choose one of four pre-defined capabilities:
     - **Local password login disabled**
     - **No password (unsecured)**
     - **Password set**
     - **Locked out**
   - **Password**—Enter the password.
   - **Confirm password**—Re-enter password.
2. Click **Add User** to complete the operation.
3. Click **Save** at the top of the page to make changes persistent across reboots.

Configuring AAA (Web Interface)

Configure AAA settings; accounting options are not supported at this time. RADIUS or TACACS+ authentication must be configured before these options can be specified with this command.

To configure AAA authentication:

- From the left navigation pane in the **System Config** tab, select **AAA**.
  The **AAA Authentication** page appears.

- Setting the Authentication Method List on page 308
- Authorization on page 308
- Configuring Authentication Failure Tracking on page 309

Setting the Authentication Method List

To set the authentication method order:

1. Select from the drop-down list box for the **First Method**, **Second Method**, **Third Method**, and **Fourth Method**, authentication methods. The order in which the methods are specified is the order in which they are attempted. The following authentication methods must be configured:
   - **Local**
   - **RADIUS**
   - **TACACS+**
   - **LDAP**
2. Click **Apply** to complete the operation; click **Cancel** to revert to existing configuration.
3. Click **Save** at the top of the page to make changes persistent across reboots.

Authorization

To set authorization options:

1. Select a **Map Order**— How the remote user mapping behaves when authenticating users using RADIUS or TACACS+. If the authenticated username is valid locally, no mapping is performed. Options:
   - **remote-first**— If a local-user mapping attribute is returned and is a valid local username, map the authenticated user to the local user specified in the attribute. Otherwise, if the attribute is not present or not valid locally, use the user specified as the **default-user**.
   - **remote-only** — Only try to map a remote authenticated user if the authentication server sends a local-user mapping attribute; otherwise, no further mapping is tried.
   - **local-only** — All remote users are mapped to the user specified by **Map Default User**. Any vendor attributes received by an authentication server are ignored.

2. Select a **Map Default User**—Local account that a non-local user authenticated using RADIUS or TACACS+ is logged on as; you must select a local and enabled user. This mapping is used depending on the setting of **Map Order**. Click **Apply** to complete operation; click **Cancel** to revert to existing configuration.
3. Click **Save** at the top of the page to make changes persistent across reboots.

Configuring Authentication Failure Tracking

To configure authorization failure tracking options:

1. Enter information in the following boxes:
   - **Enable authentication failure tracking**
   - **Enable lockouts based on authentication failures**
   - **Exempt 'admin' account from lockouts**
   - **Exempt unknown accounts from tracking**
   - **Lock account after consecutive auth failures**
   - **Allow retry on locked accounts (unlock time) after**
   - **Temporary lock after each auth failure (lock time) for**
2. Click **Apply** to complete the operation; click **Cancel** to revert to existing configuration.
3. Click **Save** at the top of the page to make changes persistent across reboots.

Related Documentation

- Logging In to Media Flow Controller for the First Time (Web Interface) on page 283
- System Configuration Overview (Web Interface) on page 287
- Administering Media Flow Controller Overview on page 315
- Monitoring Media Flow Controller Statistics (Web Interface) on page 284

Configuring Faults and Logging (Web Interface)

Before you configure Media Flow Controller for the first time, see "Before You Configure Media Flow Controller" on page 20.

- Configuring SNMP (Web Interface) on page 309
- Configuring Fault Reporting (Web Interface) on page 311
- Configuring System Logging (Web Interface) on page 313

Configuring SNMP (Web Interface)

- Configuring SNMP (Web Interface) on page 310
- Configuring SNMP on page 310
- Managing Trap Sinks on page 310
- Adding a New Trap Sink on page 310

Configuring SNMP (Web Interface)

Configure Simple Network Management Protocol (SNMP options).

To configure SNMP:

- From the left navigation pane in the **System Config** tab, select **SNMP**.
  The **SNMP** page appears.

Configuring SNMP

To set **SNMP Configuration**:

1. Enter the following information in the text boxes:
   - **Enable SNMP**
   - **Enable SNMP communities**
   - **Enable SNMP traps**
   - **Sys Contact**
   - **Sys Location**
   - **Read-Only Community**
   - **Default Trap Community**
2. Click **Apply** to complete fault notification configuration; click **Cancel** to revert to existing configuration.
3. Click **Save** at the top of the page to make changes persistent across reboots.

Managing Trap Sinks

To view and manage **Trap Sinks**:

1. View the following information on trap sinks:
   - **Host**
   - **Community**
   - **Version**
   - **Enabled**
2. To delete a trap sink, select a trap sink and click **Remove Trap Sink**.
3. To enable a trap sink, select a trap sink and click **Enable Trap Sink**.
4. To disable a trap sink, select a trap sink and click **Disable Trap Sink**.
5. Click **Save** at the top of the page to make changes persistent across reboots.

Adding a New Trap Sink

To add a new trap sink:

1. Enter the following information in the text boxes in the **Add New trap Sinks** area:
   - **Trap Sink IP**
   - **Community**
   - **Trap Type**
2. Click **Add Trap Sink** to complete adding the new trap sink.
3. Click **Save** at the top of the page to make changes persistent across reboots.

Configuring Fault Reporting (Web Interface)

Configure Fault Reporting (event notification) options. See **email** in the *Media Flow CLI Controller Command Reference* for CLI details.

To configure Fault Reporting:

- From the left navigation pane in the **System Config** tab, select **Faults**. The **Faults** page appears.

- Setting Fault Reporting on page 311
- Notifying Recipients on page 312
- Adding New Notify Recipients on page 312

Setting Fault Reporting

Set the SMTP server, domain name overrides, return address, and other options.

To set **Fault Reporting**:

1. Enter the following information in the text boxes:
   - **SMTP server**—Use a **hostname** or **IP address** to set the mail relay (**mailhub** in the CLI) to use to send notification e-mails.
   - **Domain name override**—Use a **hostname** or **IP address** to set the domain name from which e-mails will appear to come (provided that the return address is not already fully-qualified). This is used in conjunction with the system hostname to form the full name of the host from which the e-mail appears to come. The rules are as follows:
     a. If an e-mail domain is specified using this command, it is always used. If the **hostname** has any dots in it, everything to the right of the first dot is stripped and the e-mail domain is appended.
     b. Otherwise, if the **hostname** has dots in it, it is used as is. If not set, the currently-active system domain name is used. This can come either from the resolver configuration, or from a state dynamically instantiated by DHCP.
   - **Return address**—Set the username or fully-qualified return address from which e-mail notifications are sent. If the string provided contains an at (@) sign, it is considered fully-qualified and is used as-is. Otherwise, it is considered just the username, and Media Flow Controller appends @***hostname.domain***. The default is

do-not-reply, but this can be changed to **admin** or as desired in case something along the line does not like fictitious addresses.

2. Select or deselect the following options:
   - **Include hostname in return addr**—Include (or do not include by clearing) the hostname in the return address for e-mail notifications. This only takes effect if the return address does not contain an at (@) sign.
   - **Enable autosupport notifications**—Enable or disable (by clearing) the sending of e-mail to vendor autosupport when certain failures occur.
   - **Enable SMTP authentication**—Enable (by checking) or disable SMTP authentication for fault reporting. If enabled, also enter information in the following text boxes:
     - Username—Set a username for SMTP authentication of e-mails.
     - Password—Set a password for SMTP authentication of e-mails; if no password is set, the user is prompted for the password. The only authentication method supported is "LOGIN," which sends the password in the clear (base64); so users should be aware that this involves some security risk.
3. Click **Apply** to complete Fault notification configuration; click **Cancel** to revert to existing configuration.
4. Click **Save** at the top of the page to make changes persistent across reboots.

Notifying Recipients

To view and delete recipients for stats alarms:

1. View the following information on configured **Notify Recipients**:
   - **Email**—Address of the configured recipient.
   - **Detail**—Whether or not this recipient is sent detailed or summarized e-mails. Each e-mail potentially has both a detailed and summarized form, where the detailed form has a superset of the information.
   - **Infos**—Whether or not this recipient receives **Info** class e-mails.
   - **Failures**—Whether or not this recipient receives **Failure** class e-mails.
2. To delete a recipient, select a notify recipient and select **Remove Recipients**.
3. Click **Save** at the top of the page to make changes persistent across reboots.

Adding New Notify Recipients

Add recipients for e-mail notifications of fault events.

To add new notify recipients:

1. Enter the **Email Address** address of the recipient in the **Add New Notify Recipients** area.
   .
2. Select or deselect these options:

- **Get Detail**—Select to send this recipient detailed or summarized e-mails.
- **Get Infos**—Select to send this recipient **Info** class e-mails.
- **Get Failures**—Select to send this recipient **Failure** class e-mails.

3. Click **Add Recipient** to complete adding the new notify recipient.
4. Click **Save** at the top of the page to make changes persistent across reboots.

Configuring System Logging (Web Interface)

Configure logging options. See"Configuring Media Flow Controller System Log" on page 360 for more information.

These settings configure the system log (syslog) that records all system activity such as user logins, configuration changes, and system condition changes. It does not record service activity or errors. The Media Flow Controller errorlog records service related errors but is mostly useful for debugging by Juniper Networks Support. Media Flow Controller provides several service-specific logs.

To configure logging options:

- From the left navigation pane in the **System Config** tab, select **Logging**. The **Logging** page appears.

Setting Local Log Filtering

To set local log filtering in the **Local Log Filtering** area:

1. Select a **Minimum severity level**:
   - **None**—Media Flow Controller does not log anything from this class.
   - **Emerg**—System is unusable or cannot recover.
   - **Alert**—Action must be taken immediately for functioning to continue.
   - **Critical**—An unexpected error-causing condition or response for unknown reasons.
   - **Error**—Error conditions.
   - **Warning**—An anomalous condition that can be ignored and functioning continue, but may affect operations.
   - **Notice**—Normal but significant condition or response that could affect operations (default).

- **Info**—Normal but significant condition or response that does not affect operations.
- **Debug**—Messages generated by the system debugging utility.

2. Click **Apply** to complete Log filtering configuration, click **Cancel** to revert to existing configuration.
3. Click **Save** at the top of the page to make changes persistent across reboots.

Setting Local Log Rotation

Set log rotation parameters; this is especially valuable if this Media Flow Controller will be managed by Central Management Console.

To set **Local Log Rotation** options:

1. Select or deselect the following options:
   - **Rotate every**—**Day** (at midnight), **Week** (first day, at midnight), or **Month** (first day, at midnight).
   - **Rotate when log reaches**—A certain size. The file size is checked hourly, so if it passes the threshold in the middle of the hour it is not rotated right away.
   - **Rotate when log reaches**—A percentage of storage (/var) space. The var size is checked hourly, so if it passes the threshold in the middle of the hour it is not rotated right away.
2. Enter this information:
   - **Keep at most *n* log files**—How many logs to maintain on the system. If the number of log files exceeds this number (at rotation time, or when this setting is lowered), the system deletes as many as necessary, starting with the oldest, to bring it down to this number.
3. Click **Apply** to complete Log rotation configuration; click **Cancel** to revert to existing configuration: or click **Force Rotation** to immediately generate a system log file and start logging over.
4. Click **Save** at the top of the page to make changes persistent across reboots.

Viewing Remote Log Sinks

To view and delete configured log sinks (remote servers receiving log messages from this system):

1. View the following information:
   - **Remote Sink**—Address of configured remote sink.
   - **Minimum Severity**—The configured log severity level for this remote sink.
2. To delete a log sink, select a log sink and click **Remove**.
3. Click **Save** at the top of the page to make changes persistent across reboots.

Adding New Remote Sinks

Add new remote sinks (remote servers receiving log messages from this system).

To add a new remote sink:

1. Enter an **IP address**.
2. Select a **Minimum Severity** level (described in "Setting Local Log Filtering" on page 313).
3. Click **Apply** to add the new remote sink; click, **Cancel** to revert to existing configuration.
4. Click **Save** at the top of the page to make changes persistent across reboots.

Configuring Log Format

To set a log format:

1. Select either **Standard** (default) or Web trends Enhanced Log Format (**WELF**) on the **Log Format** area. If you select **WELF**, a **WELF firewall name** option appears; specify the firewall name that should be associated with each message logged in WELF format. If no firewall name is set, the hostname is used by default.
2. Click **Apply** to set the format; click **Cancel** to revert to existing configuration.
3. Click **Save** at the top of the page to make changes persistent across reboots.

Related Documentation

- Logging In to Media Flow Controller for the First Time (Web Interface) on page 283
- System Configuration Overview (Web Interface) on page 287
- Administering Media Flow Controller Overview on page 315
- Monitoring Media Flow Controller Statistics (Web Interface) on page 284

Administering Media Flow Controller Overview

You can perform standard Media Flow Controller administrative tasks using the Web interface, also known as the Management Console.

- Managing Configuration Files (Web Interface) on page 316
- Installing Licenses (Web Interface) on page 318
- Restarting Services on page 318
- Upgrading the System (Web Interface) on page 319
- Rebooting the System (Web Interface) on page 320
- Configuring the Web Interface (Web Interface) on page 321

Related Documentation

- Logging In to Media Flow Controller for the First Time (Web Interface) on page 283
- Administering Media Flow Controller Overview on page 315

- Monitoring Media Flow Controller Statistics (Web Interface) on page 284
- About the Media Flow Controller Web Interface on page 281

Managing Configuration Files (Web Interface)

The system can store one or more configuration files on persistent storage with one of the files is designated as **active**: the file that configuration is loaded from on boot, and to which configuration is saved upon a save request. Configuration changes are immediately applied to the running configuration, but are not made persistent until they are explicitly saved.

To manage configuration files:

- From the left navigation pane in the **System Config** tab, select **Config Mgmt**. The **Configurations** page appears.

- Managing Configuration Files on page 316
- Managing the Active Configuration on page 316
- Uploading a Configuration File on page 317
- Executing CLI Commands on page 317
- Importing a Configuration on page 317

Managing Configuration Files

Select a configuration and take action.

To manage configuration files from the **Configurations** page:

1. View the following information for each available configuration file:
   - **Filename**—Name of the configuration.
   - **Active**—Whether or not the configuration is currently in use.
2. Select a file using the check boxes, and:
   - Click the filename link to open a printout of that configuration file.
   - Click the **view running config** link to open a printout of the running configuration.
   - Click **Delete** to remove from the system the selected configuration file.
   - Click **Switch-To** to make the selected configuration active.
   - Click **Download** to download the selected configuration as a binary file; you are given the option of opening the file or saving it.
3. Click **Save** at the top of the page to make changes persistent across reboots.

Managing the Active Configuration

For the current, active configuration, take various actions.

To manage the **Active Configuration** from the **Configurations** page:

1. Use the action buttons:
   - Click **Save** to save the active configuration file
   - Click **Revert** to discard the running configuration and apply the active configuration.
   - Click **Reset** to reset both running and active configuration files to the factory defaults.
   - Click **Save As** to save the active configuration as a new file; enter a name in the **New filename** text box.
2. Click **Save** at the top of the page to make changes persistent across reboots.

Uploading a Configuration File

Upload your configuration as local binary file or a local text file from the **Configurations** page:

1. Click the box for the local file you want to upload, either the binary file or the text file.

   If you select the binary file, it is saved as a separate file; if you select the text file, it is immediately executed in the running configuration.
2. Select **Browse** to locate the file.
3. Click **Upload** to upload the configuration file.

   The text file will begin executing immediately, while the binary file is saved as a separate file.

Executing CLI Commands

Use this text box to enter CLI commands, each on a separate line, to be executed ad-hoc. End with the **write memory** command. When done, click **Execute CLI commands**.

Importing a Configuration

Retrieve a configuration from a remote system.

To import a configuration:

1. Enter the following information in the text boxes:
   - **Hostname or IP address**—The location of the desired configuration.
   - **Remote Username**—Login username.
   - **Remote Password**—Login password for the set **Remote Username**.
   - **Remote Config Name**—Filename of the desired configuration.
   - **New Config Name**—A new name for the imported configuration (optional).
   - **Import shared data only**—Uncheck to import all nodes, even those not available on this system.

- **File Transfer Protocol**—Select either **SCP (Secure Copy Protocol via SSH)** or **SFTP (SSH File Transfer Protocol)**.
- **Service Port**—Enter the number of the service port you want to use.

2. Click **Import Configuration**.

Related Documentation

- Installing Licenses (Web Interface) on page 318
- Administering Media Flow Controller Overview on page 315
- Logging In to Media Flow Controller for the First Time (Web Interface) on page 283

Installing Licenses (Web Interface)

Use this page to view and remove installed licenses, and add new licenses.
To manage licensing:

- From the left navigation pane in the **System Config** tab, select **Licensing**. The **Licensing** page appears.

- Viewing and Deleting Installed Licenses on page 318
- Adding New Licenses on page 318

Viewing and Deleting Installed Licenses

View and delete installed licenses.

To delete a license from the **Licensing** page:

1. Select a license and click **Remove**.
2. Click **Save** at the top of the page to make changes persistent across reboots.

Adding New Licenses

Use this area to manually enter a license key.

To manually add licenses from the **Licensing** page:

1. Enter one or more licenses on separate lines into the text box and click **Add Licenses**.
2. Click **Save** at the top of the page to make changes persistent across reboots.

Related Documentation

- Managing Configuration Files (Web Interface) on page 316
- Administering Media Flow Controller Overview on page 315
- Logging In to Media Flow Controller for the First Time (Web Interface) on page 283

Restarting Services

Several services require restart after making changes.

When any changes are made to delivery protocols or network settings, you must restart the **mod-delivery** service; use **mod-ftp** for changes made to FTP settings, and **mod-log** for changes to logging settings. The **mod-oom** service (offline origin fetch manager) is provided for debugging purposes only.

To restart services:

1. From the left navigation pane in the **System Config** tab, select **Restart Services**. The **Restart Services** page appears.
2. Select a **Service Name** and click **Restart**.

Related Documentation

- Installing Licenses (Web Interface) on page 318
- Upgrading the System (Web Interface) on page 319
- Rebooting the System (Web Interface) on page 320
- Configuring the Web Interface (Web Interface) on page 321
- Viewing Logs Overview on page 322

Upgrading the System (Web Interface)

Use this page to install upgrades.
To install upgrades:

- From the left navigation pane in the **System Config** tab, select **Upgrade**. The **Upgrades and Imaging** page appears.

- Managing Installed Images on page 319
- Installing a New Image on page 319

Managing Installed Images

To manage installed images from the **Upgrades** page:

1. View the images installed on the two boot partitions.
2. Click **Switch Boot Partition** if the image you want to install is in the other partition.
3. Click **Save** at the top of the page to make changes persistent across reboots.

Installing a New Image

To set partition, use **Switch Boot Partition**, above, if needed.

To install a new image from the **Upgrades** page:

1. Select the appropriate radio buttons and enter the following information:
   - **Install from URL**—Enter the URL and file path of the install image.

- **Install using SCP or SFTP**—Enter the **URL** and file path of the install image and a **Password** allowing access.
- **Install from local file**—Use the **Browse** button to locate the file on your local system.

2. Select installation options:
   - **View image upgrade process**— When this is checked, you get a progress bar and status messages of the upgrade process.
   - **Image validation**; select from the drop-down list box:
     - Validate if signature is present (default value)
     - Require signature and validate
     - Ignore signature
3. Click **Install Image**. A progress bar appears showing the image install.
4. To complete the upgrade, go to the **Reboot** page and reboot the system.
5. Click **Save** at the top of the page to make changes persistent across reboots.

Related Documentation

- Installing Licenses (Web Interface) on page 318
- Restarting Services on page 318
- Rebooting the System (Web Interface) on page 320
- Accessing Tech Support (Web Interface) on page 321
- Viewing Logs Overview on page 322

Rebooting the System (Web Interface)

A reboot brings up the same configuration that was last active.
To reboot or shut down the appliance:

- From the left navigation pane in the **System Config** tab, select **Reboot**.
  The **System Reboot or Shutdown** page appears.
- Rebooting or Shutting Down the System on page 320

Rebooting or Shutting Down the System

Rebooting the system brings up the last active configuration.

To reboot or shut down the system:

1. Click **Reboot** to bring up the last active configuration.
2. Click **Shutdown** to turn the system off.

Related Documentation

- Restarting Services on page 318
- Installing Licenses (Web Interface) on page 318

- Accessing Tech Support (Web Interface) on page 321
- Viewing Logs Overview on page 322

Accessing Tech Support (Web Interface)

To access tech support from the Web interface:

- From the left navigation pane in the **System Config** tab, select **Tech Support**.
  The **Tech Support** page appears.

- Tech Support on page 321

Tech Support

To access Tech Support:

- Click **Download** in the **Tech Support Report** area to access the tech support reports.
- Select the section you want from the **Snapshots** menu, and click **Download** or **Delete**.

Related Documentation

- Rebooting the System (Web Interface) on page 320
- Restarting Services on page 318
- Upgrading the System (Web Interface) on page 319
- Viewing Logs Overview on page 322

Configuring the Web Interface (Web Interface)

- Configuring the Web Interface (Web Interface) on page 321
- Configuring the Web Interface Proxy (Web Interface) on page 322

Configuring the Web Interface (Web Interface)

Configure Media Flow Controller Web interface options. Before you configure the Media Flow Controller Web interface, see "Before You Configure Media Flow Controller" on page 20.

To configure the Web interface Management Console:

- From the left navigation pane in the **System Config** tab, select **Web**.
  The **Web Settings** page appears.

To set parameters for the Media Flow Controller Web interface:

1. Enter the following information to the text boxes:
   - **Enable Web Configuration**—Allow configurations using the Web interface.
   - **Auto Logout Timeout**—Control the length of user inactivity required before the Web interface automatically logs a user out.

- **Enable HTTP** and set an **HTTP Port**, deselect to disable HTTP.
- **Redirect HTTP to HTTPS**—Click to enable.
- **Enable HTTPS** and set an **HTTPS Port**, deselect to disable HTTPS.
- **Web Session Renewal**—Control the length of time before Web session cookies are automatically regenerated.
- **Web Session Timeout**—Configure time after which a session expires.

2. Click **Apply** to complete operation; click **Cancel** to revert to existing configuration. You can also click **Generate New HTTPS Certificate**.
3. Click **Save** at the top of the page to make changes persistent across reboots.

Configuring the Web Interface Proxy (Web Interface)

Configure the Web interface proxy, if needed.

To set parameters for the Media Flow Controller Web interface when proxied:

1. Enter the following information to the text boxes:
   - **Web Proxy address**—Specify a proxy to be used for any HTTP or FTP downloads.
   - **Web Proxy port**—If no port is specified, the default is **1080**.
   - **Authentication type**—Configure the type of authentication to be used with a Web proxy; either **None** or **Basic**.
   - **Basic auth username**—If you are authenticating with Basic HTTP authentication, enter a username.
   - **Basic auth password**—If you are authenticating with Basic HTTP authentication, enter a password for your **Basic auth username**.
2. Click **Apply** to complete operation; click **Cancel** to revert to existing configuration.
3. Click **Save** at the top of the page to make changes persistent across reboots.

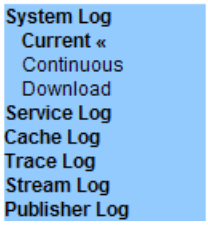
Related Documentation

- Rebooting the System (Web Interface) on page 320
- Restarting Services on page 318
- Upgrading the System (Web Interface) on page 319
- Viewing Logs Overview on page 322

Viewing Logs Overview

Media Flow Controller supplies several logs. Figure 23 on page 323 shows the **Logs** menu.

Figure 23: Logs tab left navigation menu



NOTE: The **FMSConnector Log**, **FMSAccess Log**, and **FMSEdge Log** functionality is not supported by Media Flow Controller from version 12.2.3 onward.

Four options are available for each section:

- **Continuous** log, updated every 10 seconds.
- **Current** log, that day's activity.
- **Download** log—Opens a **File Open** dialog so you can save the trace log locally.
- **Archived** (does not display if not applicable)–Log (1 - n), past logs.

To view system log information, see:

- Viewing the System Log (Web Interface) on page 323
- Viewing the Service Log (Web Interface) on page 323
- Viewing the Cache Log (Web Interface) on page 324
- Viewing the Trace Log (Web Interface) on page 324
- Viewing the Stream Log (Web Interface) on page 324
- Viewing the Publisher Log (Web Interface) on page 324

Viewing the System Log (Web Interface)

Click **System Log** to access system logs; this log (syslog) records system activity.

Example:

```
Jan  6 00:10:00 MFC httpd: [Wed Jan 06 00:10:00 2010] [notice] Apache configured
 -- resuming normal operations
```

Fields (some may not display):

Date_and_Time System_Hostname_and_Service_Name: [Process_Date_and_Time] [Severity_Level] Event

Viewing the Service Log (Web Interface)

The Media Flow Controller service (access) log records each HTTP transaction going through Media Flow Controller; in the CLI this is the **accesslog**. See "Reading the Service Log (Access Log)" on page 340 for details, including Status Codes and Status Sub Codes.

Example:

```
# Version: 1.0
# Software: Media Flow Controller v(mfc-12.3.0-qa)
# Date: 2012/07/25 12:47:52 GMT
# Remarks: default
# Format: x-cache-hit cs-host x-remote-user user time "cs-request" sc-status
sc-bytes-content x-namespace "cs(Cache-Control)" "cs(Pragma)" "sc(Cache-Control)"
 "sc(Pragma)" "sc(Vary)" sc-substatus x-namespace x-server
```

Fields:

[Date_and_Time] [Version] [Remarks] [Format]

Viewing the Cache Log (Web Interface)

The Media Flow Controller Cache log records all cache related activity; in the CLI this is the **cachelog**. Use this tab to view Media Flow Controller caching events. See "Reading the Cache Log" on page 341 for details including event types.

Viewing the Trace Log (Web Interface)

Media Flow Controller includes a delivery trace facility to help diagnose the handling of a particular HTTP request; trace results are written to the trace log; in the CLI this is the **tracelog**. See "Reading the Trace Log" on page 347 for details including trace points.

Viewing the Stream Log (Web Interface)

This Media Flow Controller service log records RTSP streaming transactions; in the CLI this is the **streamlog**. Use this tab to access the Media Flow Controller Service log. See "Reading the Stream Log" on page 345 for details.

Viewing the Publisher Log (Web Interface)

The Publisher Log, records the publishing details of Media Flow Publisher. This log is generated by Media Flow Controller. See **streamlog** in the Media Flow Controller CLI Command Reference for details.

Related Documentation

- Viewing Reports (Interface Statistics) on page 329
- Monitoring Media Flow Controller Statistics (Web Interface) on page 284
- About the Media Flow Controller Web Interface on page 281
- *Using the Dashboard Tab Overview*

Using the Dashboard Tab Overview

The Dashboard tab provides a graphical, at-a-glance view for monitoring Media Flow Controller usage and performance information. See Figure 24 on page 325.

Figure 24: Dashboard

The Dashboard tab displays the Media Flow Controller software version at the top left under the dashboard name.

The left statistics pane displays Media Flow Controller system usage information, including the cumulative uptime, gigabytes of objects delivered, bandwidth savings, cache hit ratio, number of objects delivered, and CPU usage. See "Dashboard Tab Statistics" on page 325. The graph panes to the right display the number of open connections in each configured protocol over time, weekly bandwidth savings per day, total cache throughput in megabytes per second over time, and the cache tier throughput. See "Dashboard Tab Graphs" on page 328.

The Dashboard tab also includes a top-left menu to other dashboard pages. See "Dashboard Tab Menu Dashboards" on page 328

You can set the refresh interval in seconds and select **Pause**.

You can view the following:

Dashboard Tab Statistics

The Media Flow Controller Dashboard provides vital system usage statistic in the left pane.

A description of the Dashboard statistics follows:

- **Cumulative since**—The time that the Media Flow Controller has been running without reboot or shutdown.
- **GB delivered**—The total byte count of all objects Media Flow Controller delivered since the last restart.
- **Bandwidth Savings**—
  - **Cumulative**—Equals the bytes difference between by the HTTP delivery engine and fetched from origin divided by /total bytes delivered by HTTP delivery engine.
    - Cumulative Bandwidth Savings is calculated since the last start/restart of the HTTP delivery engine.
    - Cumulative Bandwidth Savings may be negative and always be less than (<) 100 percent
    - Cumulative Bandwidth Savings measures HTTP delivery engine only. It does not include other process traffic, such as SSH, dashboard, Content Ingest Manager, SNMP, and so on.
  - **Last 24 Hour Bandwidth Savings**—Equals the bytes difference between delivered by HTTP delivery engine and fetched from origin in the last 24 hours divided by /total bytes delivered by HTTP delivery engine in the last 24 hours.
    - Last 24 Hour Bandwidth Savings is calculated in last 24 hours of HTTP delivery engine.
    - Last 24 Hour Bandwidth Savings may be negative and always be less than (<) 100 percent.
    - Last 24 Hour Bandwidth Savings measures HTTP delivery engine only. It does not include other process traffic such as SSH, Dashboard, Content Ingest Manager, SNMP, and so on.
  - **Peak Hour (9PM) Bandwidth Savings**—Equals the bytes difference between delivered by HTTP delivery engine and fetched from origin in peak hour divided by /total bytes delivered by HTTP delivery engine in peak hour
    - Peak Hour Bandwidth Savings is calculated in the Peak Traffic Hour of the HTTP delivery engine.
    - Peak Hour Bandwidth Savings may be negative and always be less than (<) 100 percent.
    - Peak Hour Bandwidth Savings measures the HTTP delivery engine only. It does not include other process traffic, such as SSH, Dashboard, Content Ingest Manager, SNMP, and so on.
- **Cache Hit ratio**—The number of objects Media Flow Controller served from RAM or disk divided by /the total number of objects served.
  - **Bytes (Cumulative) Cache Hit Ratio**— Equals the total bytes delivered to client from RAM or disk cache divided by /total bytes delivered to client.

- Bytes (Cumulative) Cache Hit Ratio is calculated since the last start or restart of the HTTP delivery engine.
- Bytes (Cumulative) Cache Hit Ratio is always greater than equal to (>=) 0 percentand less than (<) 100 percent.
- Bytes (Cumulative) Cache Hit Ratio measures the HTTP delivery engine only. It does not include other traffic, such as SSH, Dashboard, Content Ingest Manager, SNMP, and so on.
- If the Media Flow Controller HTTP delivery engine is restarted, the Cache Hit Ratio retains the value and is not reset.

- **Bytes (Last 24–Hour) Cache Hit Ratio** —Equals the total bytes delivered to client from RAM or disk in the last 24 hours divided by /total bytes delivered to client in the last 24 hours.
  - Bytes (Last 24–Hour) Cache Hit Ratio is calculated in the last 24 hours data delivered by the HTTP delivery engine.
  - Bytes (Last 24–Hour) Cache Hit Ratio will always be less than equal to (<=) 0 percent and greater than (<) 100 percent.
  - Bytes (Last 24–Hour) Cache Hit Ratio measures the HTTP delivery engine only. It does not include other process traffic, such as SSH, Dashboard, Content Ingest Manager, SNMP, and so on.
  - If the Media Flow Controller HTTP delivery engine is restarted during the last 24 hours, the Cache Hit Ratio carries over the value and does not reset it.
- **Objects (Cumulative) Cache Hit Ratio**—Equals the total objects delivered to client from RAM or disk.
  - Objects (Cumulative) Cache Hit Ratio is calculated since the last start/restart of HTTP delivery engine.
  - Objects (Cumulative) Cache Hit Ratio will always be greater than equal to (>=) 0 percent and less than (>) 100 percent.
  - Objects (Cumulative) Cache Hit Ratio measures the HTTP delivery engine only. It does not include other process traffic, such as SSH, Dashboard, Content Ingest Manager, SNMP, and so on.
  - If the Media Flow Controller HTTP delivery engine is restarted during the last 24 hours, the Cache Hit Ratio carries over the value and does not reset it.
- **Objects (Last 24 Hour) Cache Hit Ratio**—Equals the total objects delivered to the client from RAM or disk in the last 24 hours divided by /the total number of objects served to users from Media Flow Controller.
  - Objects (Last 2 Hours) Cache Hit Ratio us calculated in the Last 24 Hours data delivered by HTTP delivery engine.
  - Objects (Last 2 Hours) Cache Hit Ratio will always be greater than equal to (>=) 0 percent and less than (>) 100 percent.

- Objects (Last 2 Hours) Cache Hit Ratio measures the HTTP delivery engine only. It does not include other process traffic, such as SSH, Dashboard, Content Ingest Manager, SNMP, and so on.
- If the Media Flow Controller HTTP delivery engine is restarted during the last 24 hours, the Cache Hit Ratio carries over the value and does not reset it.

- **Objects Delivered**

Dashboard Tab Graphs

The Dashboard tab includes four graphs that display Media Flow Controller performance:

- **Open Connections**—Displays the Media Flow Controller connections to the client, both HTTP and RTSP, and the connections to the origin server.
- **Weekly Bandwidth Savings**—Displays how much traffic is served from cache, resulting in bandwidth savings.
- **Cache Throughput**—Displays the delivery bandwidth across various sources, such as RAM, disk, or origin.
- **Cache Tier Throughput**—Displays the delivery throughput across various cache tiers (RAM, SSD, SAS, and SATA). Green is served from cache, Yellow is indicates cache promotion, and Red is evicted from cache.

Dashboard Tab Menu Dashboards

- Disk Cache Dashboard on page 328
- Bandwidth Savings (Cache Hit Rate) Dashboard on page 328

Disk Cache Dashboard

The Disk Cache dashboard appears when you click Disk Cache from the Dashboard Tab menu at the top left. The Disk Cache dashboard allows you to monitor Media Flow Controller installed SSD and SAS disk performance at a glance The top Disk Throughput graph displays the read/write performance for each SSD and SAS disk name and type in Megabits per second. The bottom Disk Usage displays graph the total disk free and used disk space by disk name and type in Megabits Gigabytes. The Disk Usage disk also displays the total disk space and the percentage used and free.

Bandwidth Savings (Cache Hit Rate) Dashboard

The **Cache Hit Rate** dashboard provides an at-a-glance view to monitor:

- **Global Hourly Cache Hit Ratio for Last 24 Hours**—The top graph displays the cache hit ratio, origin bandwidth and bandwidth savings in gigabytes per second and cache hit ratio percentage sorted by time.
- **Namespace-Level Last 24 Hours Average Cache Hit Ratio**—The middle graph displays the cache hit ratio by namespace and percentage sorted by bandwidth.
- **Namespace *first namespace name only* Hourly Cache Hit Ratio for Last 24 Hours**—The bottom graph displays the system hourly cache hit ratio, origin bandwidth, and

bandwidth savings for the last 24 hours by bandwidth in gigabytes per second an cache hit ratio percentage.

Disk Cache Dashboard

The Disk Cache dashboard appears when you click Disk Cache from the Dashboard Tab menu at the top left. The Disk Cache dashboard allows you to monitor Media Flow Controller installed SSD and SAS disk performance at a glance The top Disk Throughput graph displays the read/write performance for each SSD and SAS disk name and type in Megabits per second. The bottom Disk Usage displays graph the total disk free and used disk space by disk name and type in Megabits Gigabytes. The Disk Usage disk also displays the total disk space and the percentage used and free.

Bandwidth Savings (Cache Hit Rate) Dashboard

The **Cache Hit Rate** dashboard provides an at-a-glance view to monitor:

- **Global Hourly Cache Hit Ratio for Last 24 Hours**—The top graph displays the cache hit ratio, origin bandwidth and bandwidth savings in gigabytes per second and cache hit ratio percentage sorted by time.
- **Namespace-Level Last 24 Hours Average Cache Hit Ratio**—The middle graph displays the cache hit ratio by namespace and percentage sorted by bandwidth.
- **Namespace *first namespace name only* Hourly Cache Hit Ratio for Last 24 Hours**—The bottom graph displays the system hourly cache hit ratio, origin bandwidth, and bandwidth savings for the last 24 hours by bandwidth in gigabytes per second an cache hit ratio percentage.

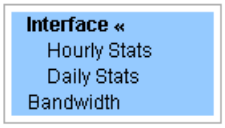
Related Documentation

- Viewing Reports (Interface Statistics) on page 329
- Configuring the Web Interface (Web Interface) on page 321
- Monitoring Media Flow Controller Statistics (Web Interface) on page 284
- About the Media Flow Controller Web Interface on page 281

Viewing Reports (Interface Statistics)

The **Reports** tab provides **Interface** and **Bandwidth** usage statistics. Figure 25 on page 329 shows the **Reports** menu.

Figure 25: Reports tab left navigation menu



To view the reports see:

- Network Usage Last 24 hours on page 330
- Interface Usage Hourly Stats on page 330

- Interface Usage Daily Stats on page 330
- Bandwidth Usage Last 24 hours on page 330

Network Usage Last 24 hours

Go to **Reports** to view network usage reports for the last 24 hours and last 7 days.

Interface Usage Hourly Stats

Go to **Reports > Interface > Hourly Stats** to view statistics for received and transmitted packets hourly for the selected interface..

Interface Usage Daily Stats

Go to **Reports > Interface > Daily Stats** to view statistics for received and transmitted packets the past 7 days for the selected interface.

Bandwidth Usage Last 24 hours

You can view bandwidth usage reports for the last 24 hours and last 7 days.

Go to **Reports > Bandwidth** to see bandwidth usage towards the client and from the origin

Related Documentation

- Viewing Logs Overview on page 322
- Configuring the Web Interface (Web Interface) on page 321
- Monitoring Media Flow Controller Statistics (Web Interface) on page 284
- About the Media Flow Controller Web Interface on page 281

CHAPTER 12

Configuring and Using Media Flow Controller Logs and Alarms

Media Flow Controller Logging Overview

The Media Flow Controller system log is used to track system-level information such as who has logged in and what commands they issued. Use the Media Flow Controller service logs, access log, cache log, error log, , fuse log, publisher log, and stream log to gather information on various aspects of the Media Flow Controller service. Use the tech-support log to send Customer Support performance information. Use the trace log to follow the path of a particular request.

System Baseline and Health

Because logging can provide vast amounts of information, finding the required information can be a daunting task. You can reduce this problem by performing a baseline audit of each installed system. A baseline audit identifies normal activity for your system, normal system log entries, normal network traffic and how the system reacts to certain conditions.

A baseline audit helps you get the most value from system log data. In particular, a baseline audit helps identify normal system behavior and in the event of a disruption or failure, it helps you determine what has changed. On a day-to-day basis, the system logs can be used to determine network health and how well the network and computer systems are running. This gives you the ability to proactively solve any issues.

Supported Logs and Log Formats

Media Flow Controller provides the following logs; reading and configuring these logs is described in this chapter:

- System logs:
  - **System log**—System events such as interface up or down, user login, configuration logging, software process crashes, and so forth. This log is not specific to the Media Flow Controller service activity. See "Reading the Media Flow Controller System Log" on page 363; see **logging** in the *Media Flow Controller CLI Command Reference* for CLI details.
  - **Cache log**—Cache activity; for example, an object has been added, deleted, or modified in the cache. See "Reading the Cache Log" on page 341; see **cachelog** in the *Media Flow Controller CLI Command Reference* for CLI details.
  - **Publisher log**—Media Flow Publisher publishing events. See "Reading the Publisher Log" on page 345; see **publisherlog** in the *Media Flow Controller CLI Command Reference* for CLI details.
- Error logs:

  - **Error log**—Media Flow Controller service-related error messages, such as cache and delivery problems; this log is for Technical Support debugging. See "Reading the Error Log" on page 344; see **errorlog** in the *Media Flow Controller CLI Command Reference* for CLI details.
  - **Debug log**—Debugging dump (sysdump). Debugging commands should be turned on only for troubleshooting specific problems or during troubleshooting sessions with technical support personnel. See **debug** in the *Media Flow Controller CLI Command Reference* for CLI details.
  - **Trace log**—A delivery trace facility to help diagnose the handling of a particular HTTP request; see . The tracing is done by logging trace points of internal steps (for example, namespace lookup, cache lookup, and so forth). See "Reading the Trace Log" on page 347; see **tracelog** in the *Media Flow Controller CLI Command Reference* for CLI details.
  - **Techsupport log**—Superset of system log information in the form of a compressed (.tgz) file. See "Reading the Media Flow Controller Tech Support Log" on page 364; see **tech-support** in the *Media Flow Controller CLI Command Reference* for CLI details.
- Access logs:
  - **Access log** (for the Media Flow Controller service)—W3C/NCSA-compliant; tracks all media requests. Media Flow Controller also allows administrators to customize the format of the access log entries. See "Reading the Service Log (Access Log)" on page 340; see **accesslog** in the *Media Flow Controller CLI Command Reference* for CLI details.
  - **Fuselog**—RTMP transaction details including what URIs are accessed and how many bytes are returned by the FUSE module. This log is generated by Media Flow Controller. See **fuselog** in the *Media Flow Controller CLI Command Reference* for CLI details.
  - **Stream log**—RTSP streaming transactions. See "Reading the Stream Log" on page 345; see **streamlog** in the *Media Flow Controller CLI Command Reference* for CLI details.

Related Documentation

- Configuring Media Flow Controller Service Logs Overview on page 333
- Reading Media Flow Controller Service Logs on page 340
- Secure Log Export Overview on page 392
- Media Flow Controller Log Codes and Sub-Codes on page 374

Configuring Media Flow Controller Service Logs Overview

- Error Log and Publisher Log Level Options on page 336
- Error Log Module Options on page 337

About Log Rotation

All Media Flow Controller service logs offer rotation options. Log rotation occurs at the configured rotation interval or frequency irrespective of whether there is traffic or not. The log file is rotated when the rotation interval is reached.

NOTE: Setting the "on-the-hour" log option to 1 sets the log rotation in hours (minimum is one hour), whereas setting the "rotate time-interval" log option to 1 sets the log rotation in minutes (minimum is five minutes). Additionally, if you set **on-the-hour enable** and **filesize**, the **filesize** setting takes precedence; should this occur between hours, the log is also rotated on the hour (if active).

Using Access Log Copy with SFTP

Media Flow Controller can auto-upload log files after they reach a certain threshold size using **accesslog copy** with **scp** or **sftp**.

To verify, use the **upload** command to manually transfer an access log file to a target host:

```
(config) # upload accesslog <all|current> <log-template-name> <sftp-location>
sftp> cd /tmp
sftp> put access2.log access2.log.tmp
```

By default, **scp** and **sftp** use password-based authentication which requires user input. To further automate log file auto-uploading, you can configure **scp** and **sftp** can be configured to use password-less authentication based on SSH public keys described in "Using SSH in Automated Scripts (CLI)" on page 24.

Access Log Format Options

For the current access log formatting options, see the *Media Flow Controller CLI Command Reference* under **accesslog**.

When you configure the access log's format specifiers, they should be enclosed in double quotes in the CLI in the following circumstances:

- When you want to customize the format of the field that is logged in the access log. (For example, **"%h foo=%r"**. The string "foo=" is printed before the request line.).
- When you want the field in access log to be enclosed within double quotes (**"%r"** in the CLI translates to **"GET /video/trailer.jpg HTTP/1.1"** in the access log).
- If you anticipate the value of a given field to have blank space and hence, you want that to be enclosed within double quotes in access logs. (For example, **"%{color}c"** means that the administrator is expecting **"blue yellow"** as the value of the token **"color"**. There is a space in between "blue" and "yellow" and hence the string must be enclosed within double quotes).

If you do not encounter the above scenarios, you can define the format specifiers without double quotes.

Stream Log Format Options

Stream log format options are shown in Table 33 on page 335. Any combination can be used.

Table 33: Stream Log Format Options

Field	Description	Example s
%a (avg b/w)	Average bandwidth, in bytes per second.	35390
%c (s-ip)	Destination IP address in the RTSP request.	192.155.14.14
%d (r-ip)	Origin server IP address.	123.14.14.55
%f (filelength)	Length of the stream, in seconds.	55
%h (c-ip)	Client source IP address.	207.19.19.20
%i	Player information used in the header (User-Agent).	NSPlayer
%j (timestamp)	Timestamp (seconds since Epoch) when transaction ended.	963873698.73
%l (cs-uri-stem)	URI stem accessed; the URL up to the first question mark (?).	rtsp://abc/hello.rm
%o (c-os)	Client operating system.	Windows
%p (player)	Client media player version.	RealPlayer or 7.0.1.15
%r (cs-uri)	Streaming URI (URL suffix) accessed.	rtsp://abc/hello.rm?there
%s (sc-status) (default)	The status code the server returns to the client. This reveals whether the request resulted in a successful response (codes beginning in 2), a redirection (codes beginning in 3), an error caused by the client (codes beginning in 4), or an error in the server (codes beginning in 5). See "Media Flow Controller Log Codes and Sub-Codes" on page 374 for the full list.	401
%t (localtime)	Local time, in *DD/MM/YYYY:HH:MM:SS GGGG* format (where GGGG is the time differential from GMT); indicates when the connection ended.	[03/Sep/2002:00:00:02 00000]
%u (cpu)	Client CPU type.	Pentium II
%v (c-osversion)	Client operating system version.	98
%x (transaction)	Transaction or reply code.	CLIENT_DATA_FROM_DISK
%A (action)	Action performed.	SPLIT, PROXY, CACHE, CONNECTING

Table 33: Stream Log Format Options *(continued)*

Field	Description	Example s
%B (begin) (clip-start)	Time when client started receiving the stream.	101
%C (client-id)	Unique client-side streaming identifier.	12345678
%D (dropped-packets)	Number of packets dropped.	24
%E (end) (clip-end)	Time when client stopped receiving the stream.	52007
%I	Number of bytes received in the request.	214
%K (resent-packets)	Number of packets resent to the client.	5
%L (request_protocol)	The protocol used during connection (for example, RTSP or RTMP).	RTSP
%N	Namespace name.	stream
%O	Number of bytes transmitted in the response.	412
%P	Total number of packets delivered to the client.	701
%R (transport)	Transport protocol used during connection.	RTP_UDP, RTP_TCP, RAW_UDP, RAW_TCP
%S (stream-id)	Unique server-side streaming identifier used to correlate the streaming access log and streaming details log entries pertaining to that server stream.	123890000
%T (time)	GMT time when the transaction ended.	10:33:02
%X (product)	Streaming product used to create and stream content.	WMT

Error Log and Publisher Log Level Options

The levels that can be set for the error log and the publisher log are shown in Table 34 on page 336.

Table 34: Error Log and Publisher Log Levels

Level Name (number)	Description
Severe (1)	Failure, immediate attention is required. Some Severe messages are informational only and do not indicate an error.
Error (2)	Function worked incorrectly.
Warning (3)	Function worked, but not as expected.
MSG (4)	System activity.

Table 34: Error Log and Publisher Log Levels *(continued)*

Level Name (number)	Description
DEBUG (5-7)	Progressively more detailed messages.

Error Log Module Options

The error log module names and codes that can display are shown in Table 35 on page 337.

Table 35: Error Log Modules

Module Name	Maps to Module or Modules
http	HTTP, HTTP Headers, Offline Origin Manager, Origin Manager
rtsp	RTSP, HTTP Headers, Origin Manager, Offline Origin Manager
media-cache	Media Manager, Buffer Manager, Caching Engine, File, TFM (temporary file manager: services offline origin manager for files under 10 MB stored until promoted to the first eligible disk cache), (cache) Analytics Manager, Media Promotion, GET Manager (queue consumer that loads the given object into the cache using GET and TFM PUT tasks), and Disk Manager
network	Network, CP (connection pooling), Tunnel Delivery, TCP (Transport Control Protocol)
nfs	NFS (network file system)
mfp-file	Media Flow Publisher File Manager
mfp-live	Media Flow Publisher Live Streaming manager
cluster	Cluster manager
auth-manager	EM (encryption manager), SSL (Secure Sockets Layer)
virtual-player	SSP (server side player manager), VPE (Video Processing Engine)
all	All modules

Related Documentation

- Media Flow Controller Logging Overview on page 332
- Reading Media Flow Controller Service Logs on page 340
- Secure Log Export Overview on page 392
- Media Flow Controller Log Codes and Sub-Codes on page 374

Configuring Media Flow Controller Service Logs (CLI)

Most Media Flow Controller logs allow you to enable or disable, copy (automatically), set the file size (for auto-copy), set the filename, set the format/field options, and set system log replication. See "Configuring Media Flow Controller System Log" on page 360 for details on system log options. Unless otherwise indicated, this procedure applies to the access log, cache log, error log, , fuse log, publisher log, stream log, and trace log ("<*>log" = any service log).

TIP: You can schedule automatic uploads of completed logs with the **<*>log copy** command. A completed log is one that has reached its set **filesize** (the default is 100). Only completed logs can be uploaded with **<*>log copy**; to see logs more frequently, you can reduce the maximum filesize with the **<*>log filesize 1** command. In that way, when a log reaches 1 MB (rather than the default 100 MB) it uploads automatically to the set URL.

To configure service logs:

1. Disable the log (these logs are enabled by default) and enable the log again.

   no <*>log enable
   <*> log enable

2. Change the default log name, ***logfile.#.yyyymmdd_hour:min:sec***, numbered sequentially by creation time.

   <*>log filename *new_name*.log

3. Optionally, change the log format (access log and stream log only). See "Access Log Format Options" on page 334 and "Stream Log Format Options" on page 335 for details.

   <*>log format <field1 field2 ...>

4. Optionally, change the log **level** and/or **module** only for error log and publisher log. The **level** determines how many messages are logged; the higher the number (**7** is highest) the more messages. The default is **1**, only severe events are recorded. The **module** focuses the log, the default is **all**. If you set **module** options, error log only records for those modules. See "Error Log Module Options" on page 337 and "Error Log and Publisher Log Level Options" on page 336 for CLI details.

   errorlog level 2
   errorlog module http
   errorlog module rtsp
   errorlog module media-cache

5. Optionally, set automatic rotation of service logs based on the log **filesize** or **time-interval**. If you set the **copy** option, when the log reaches its set **filesize** (in megabytes), or the set **time-interval**, it is copied to the specified machine using SCP (an SCP server must be installed on the target machine). First set the **copy** destination, then set the **rotate** criteria. If you do not set a **copy** destination, the rotated log is deleted.

```
<*>log copy SCP
<*>log rotate (filesize integer | time-interval minutes)
```

6. Alternatively, enable hourly log rotation; this is the same as setting **rotate time-interval 60**. If on-the-hour is enabled, it takes precedence over any **rotate** criteria configuration.

   ```
   <*>log on-the-hour (disable | enable)
   ```

7. Optionally, merge log entries with system log entries, making them available through the Web interface **System Logs** page.

   ```
   <*>log syslog replicate enable
   ```

8. Restart the log service.

   ```
   service restart mod-log
   ```

9. Upload the current service log.

   ```
   upload <*>log (current | all) <scp://
     username[:password]@hostname path
   ```

Example using the **accesslog** commands:

```
MFC (config) # no accesslog enable
MFC (config) # accesslog enable
MFC (config) # accesslog file-name sv05accesslog.log
MFC (config) # accesslog format %h %V %u %t %s %b %N
MFC (config) # accesslog copy scp://joe@sv01/home/joe
Password: *******
MFC (config) # service restart mod-log
MFC (config) # upload accesslog current scp://joe@sv01/home/joe
Password: *******
MFC (config) #
```

To make these configurations using the Web interface, go to the **Service Config** tab, **<*>log** page. After making changes, use the **EZconfig** page **Service Restart** area to restart the appropriate log service. Be sure to click **Apply** at each section and **Save** at the top right of the page to make the changes persistent across reboots or restarts.

For CLI details on the service logs, see **accesslog**, **cachelog**, **errorlog**, , **fuselog**, **streamlog**, and **tracelog** in the *Media Flow Controller CLI Command Reference*.

Related Documentation

- Reading Media Flow Controller Service Logs on page 340
- Configuring Media Flow Controller Service Logs Overview on page 333
- Media Flow Controller Logging Overview on page 332
- Secure Log Export Overview on page 392
- Media Flow Controller Log Codes and Sub-Codes on page 374

Reading Media Flow Controller Service Logs

NOTE: The Web interface name for the log is given, and the command-line interface (CLI) name is given in parentheses.

Reading the Service Log (Access Log)

The access log, referred to as the Service Log in the Web interface, records each HTTP transaction going through Media Flow Controller. This log identifies HTTP traffic. Media Flow Controller supports NCSA or custom log formats.

See "Configuring Media Flow Controller Service Logs (CLI)" on page 338 for implementation details, including how to upload the log.

See **accesslog** in the *Media Flow Controller CLI Command Reference* for CLI details.

The access log (shown as **Service Log** in the Web interface) provides the following information, in the order shown (by default, the order changes if you change the access log format). Unavailable information is indicated by a dash (-).

NOTE: Double-quote and backslash characters found in these—%r %U, %C, %V, %o, %i, %f, %u, %v, and %S—access log fields are escaped with a backslash.

- (%h) Client source IP address (IP address of the remote host)
- (%V) HTTP host (server name)
- (%u) Remote user
- (%t) Timestamp
- (" %r") Request line, in quotes
- (%s) Status code (see Table 40 on page 374)
- (%b) Bytes out
- (" %{Referrer}i"), Referrer, in quotes; the site the client reports having been referred from

- (" %{User-Agent}i" , Agent, in quotes, the identifying information that the client browser reports about itself
- (%y) Status sub-code (see Table 41 on page 377)
- (%c) Cache hit indicator
- (%A) Request in time
- (%B) First byte out time
- (%F) Last byte out time
- (%M) Data out, in milliseconds

Example:

```
10.15.2.211 10.15.2.14 - [15/Nov/2010:23:50:38 +0000] "GET /user/2M HTTP/1.0" 200
 2097152 "-" "Wget/1.10.2 (Red Hat modified)" 0 Origin 1289865038.272
1289865038.288 1289865038.334 46
10.15.2.211 10.15.2.14 - [15/Nov/2010:23:51:00 +0000] "GET /user/8M HTTP/1.0" 200
 8388608 "-" "Wget/1.10.2 (Red Hat modified)" 0 Origin 1289865060.345
1289865060.365 1289865060.640 275
10.15.2.211 10.15.2.14 - [15/Nov/2010:23:52:14 +0000] "GET /user/8M HTTP/1.0" 200
 8388608 "-" "Wget/1.10.2 (Red Hat modified)" 0 Buffer 1289865134.735
1289865134.735 1289865134.785 50
```

When a media player requests an ASX file before it is generated by the parent/edge Media Flow Controller, the system logs a 404 error code in the access log, as for all other HTTP requests.

Reading the Cache Log

Media Flow Controller supplies a log of cache activity; for example, an object that has been added, deleted, or modified in the cache.

See "Configuring Media Flow Controller Service Logs (CLI)" on page 338 for implementation details, including how to upload the log.

To upload a cache log, run **upload cachelog scp:*//URL***. You can then access the cache log at the specified system, move it (if needed,) and view its contents.

The cache log supplies the following information:

- Date and timestamp
- Event string, can include events and other information

For each type of event, **Add**, **Attribute Update**, and **Delete**; a single entry is written:

Table 36: Events and entries

Event	Entry
ADD	• Date, in square brackets • Type (**ADD**) • URI name, in double quotes • Cache tier name • Cache name • Content length, in bytes • Expiry time for this URI, in UTC format with square brackets
ATTR_UPDATE (Attribute Update)	• Date, in square brackets • Type (**ATTR_UPDATE**) • URI name, in double quotes • Cache tier name • Cache name • Expiry time for this URI, in UTC format with square brackets
DELETE	• Date, in square brackets • Type (**DELETE**) • URI name, in double quotes • Cache tier name • Cache name

Example:

```
[Fri Jun 26 19:37:09.754 2009] ADD "/http-cl18:ed239a85/100k-files/117/29" SAS
dc_3 32768 [Fri Jun 26 19:38:14 2009]
[Fri Jun 26 19:38:23.257 2009] ATTR_UPDATE "/http-cl18:ed239a85/100k-files/588/45"
 SAS dc_3 [Fri Jun 26 19:39:28 2009]
[Fri Jun 26 19:50:59.072 2009] DELETE "/http-cl18:ed239a85/100k-files/381/69" SAS
 dc_4
```

NOTE: Attribute update (ATTR_UPDATE) is performed with a namespace origin-request cache-revalidation on an individual URI. If this namespace option is set to deny, no Attribute Update entries occur. Each URI has its own expiry time. If the revalidation indicates that a URI is still valid, the attribute update event updates the URI expiry time; this makes a log entry.

The system logs a message in the cache log whenever a content ingest manager HTTP crawl instance begins and again when that instance completes. Additionally, the system logs all the generated ASX files in the cache log, as well all the cache-miss fetch requests of ASX files.

Reading the Crawler Log

The crawler log contains information that tracks the progress of the crawl operation, including when the crawl operation started, when it ended, the objects that were ingested to the cache, and the objects that were deleted from the cache.

See "Configuring Media Flow Controller Service Logs (CLI)" on page 338 for implementation details, including how to upload the log.

The crawler log provides the following information:

- **Date/Time**—Timestamp when the object was ingested into the Media Flow Controller cache or deleted from the cache, that is, when the ADD or DELETE operation occurred.
- **Operation (ADD/DELETE)**—The operation that was performed:
  - **ADD**—Ingestion of object into the cache because the object was newly published at the origin server or because the older object at the origin server was replaced by a newer instance.
  - **DELETE**—Deletion of object from the cache because the object was removed at the origin server.
- **Object name**—The object that was ingested to the Media Flow Controller cache or deleted from the cache.
- **Crawler name**—Name of the crawler instance that crawled the object.
- **Namespace name/UUID**—The namespace to which the crawled object belongs. The crawler log contains the namespace name and the Universally Unique Identifier (UUID) of the namespace. UUID is an integer, which uniquely identifies a namespace.
- **Status**—Indicates the status of object ingestion to the cache or deletion from the cache. **SUCCESS** is logged when the ADD or DELETE operation is successful; otherwise, **FAILURE** is logged.
- **Internal Error code**—Displays either a zero or a non-zero value. A zero value indicates that the object ingestion or deletion was successful. A non-zero value indicates that this operation was unsuccessful and is primarily used by Juniper Networks support engineers for debugging.
- **File size**—(Applicable only for ADD operation) The size of the ingested object.
- **Expiry time**—(Applicable only for ADD operation) Indicates the expiry time of the object in seconds.

  When an object is ingested into the cache file system, the expiry time is calculated based on the refresh-interval of the crawler or based on the namespace configuration.
- **Crawl start/end time log**—Timestamp when the crawler was started or ended.

CAUTION: If a namespace **name1** is inherited into another namespace **name2**, and if Content Ingest Manager traffic hits **name2**, the Content Ingest Manager log logs **name1** as **namespace *name*** even though the current active **namespace *name*** is **name2**. There is no impact on data delivery.

Workaround: Map **name1** to **name2** while doing any analysis on the Content Ingest Manager crawl log.

Example:

```
[Wed Feb 29 08:26:00.205 2012] Crawler "test" Started
 [Wed Feb 29 08:26:01.220 2012] DELETE "http://10.10.10.1/20mb/20mb_114.html"
test dt05:f27afd87 SUCCESS 0
 [Wed Feb 29 08:26:02.236 2012] ADD "http://10.10.10.1/test1/files.sh" test
dt05:f27afd87 SUCCESS 0 106 [Wed Feb 29 08:36:00 2012]
 [Wed Feb 29 08:26:02.237 2012] ADD "http://10.10.10.1/test1/test.wmv" test
dt05:f27afd87 SUCCESS 0 10485760 [Wed Feb 29 08:36:00 2012]
 [Wed Feb 29 08:26:02.242 2012] ADD "http://10.10.10.1/test1/test%2020.wmv" test
 dt05:f27afd87 SUCCESS 0 10485760 [Wed Feb 29 08:36:00 2012]
 [Wed Feb 29 08:26:04.217 2012] Crawler "test" Completed
```

Reading the Error Log

The Media Flow Controller error log records service-related error messages, such as cache and delivery problems. This log is for Technical Support debugging. The Media Flow Controller system log records system-related messages and errors, such as user logins and CPU problems. See "Configuring Media Flow Controller System Log" on page 360 for more information.

See "Configuring Media Flow Controller Service Logs (CLI)" on page 338 for implementation details, including how to upload the log.

By default, the error log name is "error.log." If Media Flow Controller is launched from the serial console, all error messages print out on the console. If Media Flow Controller is launched from the CLI, all error messages are logged in the file /var/log/messages.

The error logs provide freeform information; including the following fields:

- Date and time, in brackets
- Media Flow Controller module and message level in the format: **module.level**, in brackets (see tables Table 35 on page 337 and Table 34 on page 336 for descriptions); the module code might not display.
- Function name and source line number (ends with a colon).
- Message, can include codes or sub-codes given in Table 40 on page 374 and Table 41 on page 377.

Example:

```
[Tue Jan  5 18:27:31.285 2010][MOD_HTTPHDRS.MSG] get_nth_list_element:1697:
find_nth_name_value() failed rv=1
[Tue Jan  5 18:27:31.285 2010][MOD_HTTPHDRS.MSG] mime_hdr_get_nth_unknown:989:
get_nth_list_element() failed rv=2
[Tue Jan  5 18:27:31.285 2010][MOD_HTTPHDRS.MSG] mime_hdr_get_unknown:930:
find_name_value_by_name() failed rv=2
```

See also "Media Flow Controller Log Codes and Sub-Codes" on page 374.

If an ongoing instance of the content ingest manager HTTP crawler has not stopped by the time configured in the refresh interval, this is logged in the error log, as well as in the system log. The system also logs an error into the error log when the total size limit for

the crawled content is reached. All disk ingestion and eviction errors are logged in the error log.

Reading the Publisher Log

The Publisher log (**mfp.log**), records the publishing details of the Media Flow Publisher application running in Media Flow Controller. This log is generated by Media Flow Controller. See **streamlog** in the *Media Flow Controller CLI Command Reference* for CLI details.

See "Configuring Media Flow Controller Service Logs (CLI)" on page 338 for implementation details, including how to upload the log.

The format of the publisher log is similar in nature to the error log. The format of each entry in this log is as follows:

- [Date/Time field]
- [Module.ErrorLevel]—Module is either MOD_MFPLIVE or MOD_MFPFILE. ErrorLevels can be DEBUG, MSG, WARNING, ERROR, or SEVERE.
- functionName:lineNumber ID
- [ID]—Session ID is the name assigned to this publishing event using the Web configuration.
- Message

Example:

The following log entry is created by MOD_MFPLIVE for a live Media Flow Publisher session named envivioLive:

```
[Wed Mar 23 19:51:00.161 2011][MOD_MFPFILE.MSG] mfp_process_pmf:183: ID[0] PMF
file name /nkn/vpe/ live/live_envivioLive620110323_19:50:59.xml
[Wed Mar 23 19:51:00.161 2011][MOD_MFPFILE.MSG] mfp_process_pmf:183: ID[0] PMF
file name /nkn/vpe/live/live_envivioLive620110323_19:50:59.xml[Wed Mar 23
19:51:00.236 2011][MOD_MFPLIVE.MSG] addEvent:447: ID[DISP] Adding event: 1
[Wed Mar 23 19:51:00.236 2011][MOD_MFPLIVE.MSG] registerEntity:170: ID[DISP]
Register success
[Wed Mar 23 19:51:00.236 2011][MOD_MFPLIVE.MSG] processData:560:
ID[envivioLive620110323_19:50:59]  Started Receiving Data, searching for PAT,
PMT, VPID, APID
[Wed Mar 23 19:51:00.236 2011][MOD_MFPFILE.MSG]
mfp_live_start_listener_session:176:  ID[envivioLive620110323_19:50:59] Session
Started Successfully
```

The log takes a user through the life cycle of a file or live publishing request.

Reading the Stream Log

This log records RTSP streaming transactions. See **streamlog** in the *Media Flow Controller CLI Command Reference* for CLI details.

See "Configuring Media Flow Controller Service Logs (CLI)" on page 338 for implementation details, including how to upload the log.

The stream log provides the information you configure with the **streamlog format** ***format_options*** command (see "Stream Log Format Options" on page 335); by default:

- **%h**—Client source IP address
- **%c**—Destination IP address
- **%t**—Local time
- **%x**—Transaction code, possible entries are:
  - ANNOUNCE
  - BROADCAST_PAUSE
  - DESCRIBE
  - DESCRIBE_RESP
  - GET_PARAMETER
  - OPTIONS
  - PAUSE
  - PLAY
  - PLAY_RESP
  - RECORD
  - REDIRECT
  - SET_PARAMETERS
  - SETUP
  - SETUP_RESP
  - TEARDOWN
  - TEARDOWN_RESP
  - UNKNOWN
- **%r**—Streaming URI (URL suffix)
- **%s**—Status
- **%I**—Number of bytes received
- **%O**—Number of bytes transmitted

Example:

```
10.157.42.147 10.157.42.145 [23/Sep/2010:17:39:12 +0000] OPTIONS
"rtsp://10.157.42.145/rtsp/sanity_video/sample_h264_1mbit.mp4" 415 139 136
10.157.42.147 10.157.42.145 [23/Sep/2010:17:39:47 +0000] OPTIONS
"rtsp://10.157.42.145/rtsp/sanity_video/sample_h264_1mbit.mp4" 200 139 169
10.157.42.147 10.157.42.145 [23/Sep/2010:17:39:47 +0000] DESCRIBE
"rtsp://10.157.42.145/rtsp/sanity_video/sample_h264_1mbit.mp4" 200 165 862
10.157.42.147 10.157.42.145 [23/Sep/2010:17:39:47 +0000] SETUP
"rtsp://10.157.42.145/rtsp/sanity_video/sample_h264_1mbit.mp4/trackID=3" 200 203
 275
```

```
10.157.42.147 10.157.42.145 [23/Sep/2010:17:39:47 +0000] SETUP
"rtsp://10.157.42.145/rtsp/sanity_video/sample_h264_1mbit.mp4/trackID=4" 200 233
 275
10.157.42.147 10.157.42.145 [23/Sep/2010:17:39:47 +0000] PLAY
"rtsp://10.157.42.145/rtsp/sanity_video/sample_h264_1mbit.mp4" 200 213 391
10.157.42.147 10.157.42.145 [23/Sep/2010:17:40:57 +0000] TEARDOWN
"rtsp://10.157.42.145/rtsp/sanity_video/sample_h264_1mbit.mp4" 200 170 138
10.157.42.147 10.157.42.145 [23/Sep/2010:18:34:18 +0000] OPTIONS
"rtsp://10.157.42.145/rtsp/sanity_video/sample_h264_1mbit.mp4" 200 139 169
```

Reading the Trace Log

Media Flow Controller includes a delivery trace facility to help diagnose the handling of a particular HTTP request. See **tracelog** in the *Media Flow Controller CLI Command Reference* for CLI details.

The tracing is done by logging trace points of internal steps (for example, namespace lookup, cache lookup, and so on). A request is traced if the following conditions are true:

- The global trace flag is enabled using the CLI:

  delivery protocol http trace enable

- The HTTP request includes the **X-NKN-Trace** header; for example if using Wget:

  wget --header “X-NKN-Trace:” *URL*

The HTTP modules detect the above conditions and set the flag “HRF_TRACE_REQUEST” as well as other flags that direct each relevant module to log meaningful trace points. The trace points (as of Version) given are shown with variables that are instantiated with data before displaying:

OM=Origin Manager, **%s**=string data, **%d**=integer data, **%ld**=long integer data

The trace points are described in Table 37 on page 347

Table 37: Delivery Protocol HTTP Trace Points

Trace Point	Description
“Could not find the file in the “ "directory. Please check filename: %s errno = %d”	The requested file was not found in the specified directory.
“Could not mount file. Please check “ "filename: %s. errno = %d”	NFS; the requested file could not be mounted.
“Could not unmount old mount configuration. Please check whether someone is accessing this directory: %s”	NFS mount failed.
“End TRACE URI: %s. End NFS_stat. Success.”	NFS.
“End TRACE URI: %s. Error while doing STAT”	NFS.
“ Fast start buffer size set to %d bytes”	Server Side Player. Configured **fast–start** value.

Table 37: Delivery Protocol HTTP Trace Points *(continued)*

Trace Point	Description
"Fast start not enabled"	Server Side Player. The **fast-start** option was requested but not enforced in the virtual player.
"Hash authentication failed for virtual player: %s"	Server Side Player. Video will not be delivered.
"Hash authentication successful for virtual player: %s"	Server Side Player. Video will be delivered.
"Health probe request enabled and no cache flag is set"	Virtual Player. For this request, the fetched file will not be cached in Media Flow Controller.
"HTTP config does not exist in namespace configuration"	The requested namespace does not have HTTP delivery protocol or origin server configured.
"HTTP Response object %s\n%s"	HTTP Response object given.
"Mount failed for mount command: %s"	NFS mount failed.
"Namespace configuration does not exist"	The requested namespace parameter is undefined.
"NFS GET: Cannot get attributes for this file"	NFS. Requested file attributes missing.
"NFS GET: Cannot read content from file."	NFS. Requested content not in file.
"NFS GET: Programming error: Error getting object after stat finished."	NFS. Error getting object after stat finished.
"NFS GET: Programming error: NFS object not found in get after stat succeeded"	NFS. Object not found.
"NFS GET: Received TRACE URI: %s"	NFS. Received trace URI.
"NFS GET: TRACE URI: %s. End NFS get: Error getting uri"	NFS. Error getting URI.
"NFS GET: TRACE URI: %s. End NFS get: Error: Server busy"	NFS. Server busy.
"NFS GET: TRACE URI: %s. End NFS get: Success"	NFS. Get successful.
"No Virtual Player associated with this namespace"	Virtual Player. The requested virtual player option was unavailable because no virtual player is assigned to the requested namespace.
"Object %s bytes %ld to %ld served from %s"	NFS trace.
"OM connect (%s) failed"	Origin Manager. Socket connect to given hostname/IP address failed with the given error code.

Table 37: Delivery Protocol HTTP Trace Points *(continued)*

Trace Point	Description
“OM connect failed %s:%hu for object %s”	Origin Manager. Unable to connect to the origin server identified by the given hostname/IP address and port for the given object.
“OM connect failed %s:%hu for object %s”	Origin Manager. Socket connect to the given IP address and port failed for the given object.
“OM connecting %s:%hu for object %s”	Origin Manager. Socket connect to the given IP address and port is pending for the given object.
“OM gethostbyname_r(%s) failed”	Origin Manager. gethostbyname_r call failed for the given hostname/IP address with the given error code.
“OM http_response object %s\n%s”	Origin Manager. HTTP origin server response for given object and headers.
“OM returning %s bytes for object %s offset=%ld length=%ld”	Origin Manager. Returning cacheable/non-cacheable given bytes to the buffer manager for the given object starting at the given offset and length.
“Programming error. Could not get URI “ "postfix	NFS trace.
“Received TRACE URI: %s”	NFS trace.
“TRACE URI: %s. End NFS stat Error: URI not found”	NFS.
“Unmount problem: Stale NFS handle %s”	NFS; the given NFS mount point is not valid.
“URI prefix does not exist in namespace configuration”	The requested uri-prefix is not configured in the requested namespace.
“URL %s%s did not match any namespace”	Server Side Player. The incoming request could not be mapped to any configured namespace.
“URL %s%s matched namespace %s”	Server Side Player. The incoming request was mapped to a configured namespace.
“Virtual Player: %s of Type-%d has been invoked”	Virtual Player. Virtual player of the named type was invoked.

To upload a trace log, run **upload tracelog scp://*URL***. You can then access the trace log at the specified system, move it (if needed), and view its contents.

Related Documentation

- Configuring Media Flow Controller Service Logs Overview on page 333
- Media Flow Controller Logging Overview on page 332
- Secure Log Export Overview on page 392
- Media Flow Controller Log Codes and Sub-Codes on page 374

Configuring Media Flow Controller Access Log Profiles

Per Namespace Access Log Profiles

You can customize access logging on a per namespace basis and export log files to an external device using a profile. Access logs record all HTTP and HTTPS requests.

You can configure up to a maximum of 32 access log profiles. Access log profiles are also called as access log templates. You can associate a namespace with a single log profile only. Multiple namespaces can share a single log profile. Log records generated by multiple namespaces, but referring to a common log profile, are written to the log file specified by the profile configuration.

When Media Flow Controller is installed on a fresh system, default access log profile is created automatically for defining the log policies for namespaces that do not have a customized access log profile. The configuration of the default profile, on upgrade, inherits the settings from the previous Media Flow Controller release. Custom settings are inherited into the Media Flow Controller software image. You can modify the default profile configuration. You can not delete the default profile, even if no namespace is associated with it.

All format specifiers available in existing Media Flow software releases are supported to provide a seamless upgrade path.

A change in log format causes the current log file to close and a new one to open to write new log records, irrespective of any configured thresholds (rotation interval or maximum file size).

NOTE: **A change in log format does not necessarily cause an immediate log rotation, since records in the internal buffer must be flushed out.**

Log files, except for those currently used, are stored in standard GZIP compressed format.

Each profile, upon creation, uses some storage to save the log files. The log files are deleted, and the storage space is reclaimed when a profile is deleted. Directories are added when you create a profile. The administrator is responsible for ensuring that all log files under a profile are backed up or exported to external storage before deleting that profile. A deleted profile cannot be recovered, and all log files stored in it are lost. You cannot delete an access log profile if it is being used by a namespace.

On VXA2010/2002 Series Media Flow Engines, a 330 GB partition is reserved for logging. However, no disk space checks are performed when you create a profile. Administrators must ensure that the sum total sizes of all access log profiles do not exceed the space constraints at any given time. For instance, if you create two profiles, each limiting a file size to 10 GB using **accesslog *log-template-name* max-size *size***, the total expected size will be 600 GB. (The maximum configurable size is 100 MB.) This would exceed the log partition size, and might cause log files to be overwritten or lost. It is recommended that administrators only export log files once to an archiving device using the **accesslog *log-template-name* copy *url*** CLI command. Log files are then automatically exported to the specified location after each log rotation.

To customize Media Flow Controller access log formatting on a per namespace basis:

- Configure an access log profile using the **accesslog *log-template-name*** CLI command. See "Configuring an Access Log Profile" on page 352.
- Associate that log profile with one or more namespaces using the **namespace *file-name*** CLI command. See "Associating an Access Log Profile to a Namespace" on page 355.
- Check the log profile to ensure that you have configured it for your needs. See "Viewing Access Log Profile Information" on page 357
- Verify that the log profile is logging according to how you configured the log profile.

  show accesslog <*log-template-name*> <continuous | last>

Related Documentation

- Configuring an Access Log Profile on page 352
- Modifying the Default Access Log Profile on page 354
- Associating an Access Log Profile to a Namespace on page 355
- Viewing Access Log Profile Information on page 357
- Example: Access Log Profile Configuration on page 356

Configuring an Access Log Profile

The access log profile allows you to customize access log files. You can configure the following:

- Log profile name to associate with a namespace
- Log format string
- Object response code for which a log record is written to a file
- Maximum log object filter size to not log
- Secure-CP, Secure-FTP, or FTP URL and password to upload log files to
- Size threshold for uploads or file rotation
- Time limit for uploading or writing log files
- Log filename

Before you begin:

- For uploading access logs to a remote location, ensure that you know the URL and password to which to upload log files.

To configure an access log profile:

1. Enter the CLI configure mode.

   # **enable**

   # **configure terminal**

2. Configure access log settings.

   # **accesslog**

3. Configure a custom access log profile name.

   # **accesslog <*log-template-name*>**. Where <*log-template-name*> is the custom name of your access log profile. Log profile names must be in alphanumeric, 7-bit ASCII format only.

   A **default** access log profile is created when you upgrade or freshly install Media Flow Controller. You can modify the default access log. See"Modifying the Default Access Log Profile" on page 354.

4. Configure the access log profile format string.

   The supported log file formats include:

   - W3C Extended Default—See **accesslogs** in the *CLI Reference Guide*.
   - CLF—See **accesslogs** in the *CLI Command Reference Guide*.
   - NCSA-Combined—See **accesslogs** in the *CLI Reference Guide*.

   # **format <format>**

For example, enter **format clf**. If you do not specify a format, the log files are formated as W3C Extended.

5. Configure the object response code for which a long record is written to a file. You can specify multiple response codes entering a space between each code.

   # **object filter response-code <code>**

   For example, enter object filter response-code 302 403. This command filters all the other responses and logs only the records with 302 or 403 response codes. Only objects greater than 5000 are logged.

6. Configure the object size for which a log record is written to a file.

   The value must be an integer between 0 and 4294967295.

   # **object filter size <bytes>**

   For example, enter object filter size 5000.

7. Configure the automatic copy capability of access logs based on rotation criteria.

   # **copy remote URL (Secure-CP, Secure-FTP, FTP) [scp|sftp|ftp]://<username>:<password>**

   For example,**copy scp://user:mysecret1@cmn-log.domain/yt-pop-123**.

8. Configure the maximum file size per log file.

   Specify the file size in MB (a binary prefix that indicates multiplication by a power of 2 and the powers used are multiples of 10. One MB equals 1, 048, 576 bytes. The default is 100 MB. A log file is rotated after reaching the maximum file size.

   # **max-size 50**

9. Configure the file rotation interval per log file.

   The maximum duration should be between 5 to 60 minutes; or 0 to disable time-based rotation. The default is 15.

   # **max-duration 30**

10. Configure a custom log filename.

    # **file-name <*filename*>**

11. Exit the access log configuration.

    # **exit**

12. View your new access log profile.

    show accesslog <filename>

    ```
    Log Filename : my_access.log
    Max Filesize : 50 MiB
    Max files to hold : 10
    Max time duration : 45 Minutes
    Format Type: w3c-ext-custom
    Format : %h %V %u %t "%r" %s %b
    Auto Copy URL : scp://iser@cm-log.domain/yt-pop123
    ```

```
Object Size to Skip : 2000
Response Codes to Skip : 302 403
```

Related Documentation

- Example: Access Log Profile Configuration on page 356
- Per Namespace Access Log Profiles on page 350
- Modifying the Default Access Log Profile on page 354
- Viewing Access Log Profile Information on page 357

Modifying the Default Access Log Profile

When Media Flow Controller is installed on a fresh system, the default access log profile is created automatically for defining the log policies for namespaces that do not have a customized access log profile. The configuration of the default profile, on upgrade, inherits the settings from the previous Media Flow Controller release. Custom settings are inherited into the Media Flow Controller image. You can modify the default profile configuration.

To modify the default accesslog profile:

1. Enter the CLI configure mode.

   # **enable**

   configure terminal

2. Configure accesslog settings.

   # **accesslog default**

3. View the default accesslog.

   # **show accesslog default**

   ```
   (config accesslog default)#
    Access Log Profile: default
    Log Filename : example_logprofile.log
    Max Filesize : 100 MiB
    Max files to hold : 10
    Max time duration : 0 Minutes
    Format Type: w3c-ext-custom
    Format : %h %V %u %t "%r" %s %b %N %v
    Auto Copy URL : -Not Configured-
    Object Size to Skip : 0
    Response Codes to Skip : None
   ```

4. Configure the default options that you want to configure. See also "Configuring an Access Log Profile" on page 352).

   (config accesslog default)# **?**

   ```
   copy        Auto-copy the accesslog based on rotate criteria
   exit        Leave "accesslog my_profile" mode
   file-name   Configure filename
   format      Configure access log format string
   help        View description of the interactive help system
   ```

```
max-duration     Configure time-interval per log file
max-size         Configure maximum file size per log file
no               Clear accesslog config options
object           Configure log record specific settings
show             Display system configuration or statistics
```

5. View the default access log to ensure that your changes are present.

   # **show accesslog default**

6. If necessary, make any configuration changes.
7. Exit the access log configuration.

   # **exit**

Related Documentation

- Per Namespace Access Log Profiles on page 350
- Configuring an Access Log Profile on page 352
- Associating an Access Log Profile to a Namespace on page 355
- Example: Access Log Profile Configuration on page 356
- Viewing Access Log Profile Information on page 357

Associating an Access Log Profile to a Namespace

You can configure up to a maximum of 32 access log profiles. You can associate a namespace with a single log profile only. Multiple namespaces can share a single log profile. Log records generated by multiple namespaces, but referring to a common log profile, are written to the log file specified by the profile configuration.

To associate an access log profile to a namespace:

1. Enter CLI enable mode.

   # **enable**

2. Enter CLI configure mode.

   # **configure terminal**

3. Configure namespace settings and view the configured namespaces.

   # **namespace ?**

4. Create a namespace or modify an existing one.

   # **namespace <*name*>**

5. Bind the namespace to an access log profile.
   a. View the available accesslog profile names.

      # **accesslog ?**

   b. Bind the namespace to the access log profile you select:

```
# namespace <name> accesslog <log-template-name>
```

6. Activate the namespace.

   ```
   # status active
   ```

7. Exit the namespace configuration.

   ```
   # exit
   ```

Related Documentation

- Per Namespace Access Log Profiles on page 350
- Configuring an Access Log Profile on page 352
- Modifying the Default Access Log Profile on page 354
- Example: Access Log Profile Configuration on page 356
- Viewing Access Log Profile Information on page 357

Example: Access Log Profile Configuration

This example shows how to configure an access log profile that you can associate with a namespace.

Requirements

- Media Flow Controller Release 11.B.3 or later

Before You Begin

- For uploading access logs to a remote location, ensure that you know the URL and password to which to upload log files.

Example accesslog Profile Configuration

```
# accesslog youtube
# format "%c %h %r %n"
# object filter response-code 403 302
# object filter size 2048
# copy scp://user:mysecret1@cmn-log.domain/yt-pop-123
# max-size 5
# max-duration 5
# file-name yt-access.log
# exit
```

Related Documentation

- Configuring an Access Log Profile on page 352
- Modifying the Default Access Log Profile on page 354
- Associating an Access Log Profile to a Namespace on page 355

Viewing Access Log Profile Information

Viewing Access Log Profile Filenames

To display created access log profile names:

1. Enter CLI configure mode.

   # **enable**

   configure terminal

2. Configure access log settings.

   # **accesslog**

3. Display accesslog help

   # **accesslog ?**

   The access log profile filenames that you created are listed.

Viewing Access Log Filenames and Profile Filenames

To display access log names and profiles filenames:

1. Enter CLI configure mode.

   # **enable**

   # **configure terminal**

2. Configure access log settings.

   # **accesslog**

3. Display all access logs.

   # **show accesslog**

```
Media Flow Controller Access Log enabled : yes
 Analyzer Tool Enabled : no
Profile Name      | Log File        | Format Type    | Exprt | Filter
-------------------------------------------------------------------------------
default           | access.log      | w3c-ext-custom |   N   |   N
```

Viewing an Access Log Profiles Associated with a Namespace

To view the content of an access log associated with a namespace:

1. Enter CLI configure mode.

enable

configure terminal

2. View the contents of the namespace for which you want to see the associated access log profile.

   # show namespace <*namespace-name*>

   For example:

```
# show namespace my_namespace
Namespace: my_namespace
 Active: yes
 Precedence: 0
 Policy:
 Resource-Pool: global_pool
 AccessLog Profile: default
 Cluster Association:
  NONE
...
```

Viewing Access Log Profile Configuration

To view the contents of an accesslog profile:

1. Enter CLI configure mode.

   # enable

   # configure terminal

2. View your new access log profile.

   # show accesslog <*log-template-name*>

```
Log Filename : my_access.log
Max Filesize : 50 MiB
Max files to hold : 10
Max time duration : 45 Minutes
Format Type: w3c-ext-custom
Format : %h %V %u %t "%r" %s %b
Auto Copy URL : scp://iser@cm-log.domain/yt-pop123
Object Size to Skip : 2000
Response Codes to Skip : 302 403
```

Viewing the accesslog

To view the access log:

1. Enter CLI configure mode.

   # enable

   # configure terminal

2. View the access log.

   # show accesslog <*log-template-name*> <continuous | last>

```
# Version: 1.0
# Software: Media Flow Controller v(mfc-11.B.2)
# Start Date: 2011/12/09 19:13:35 GMT
# Remarks: default
# Format: cs-host -- time cs-request sc-status sc-bytes-content x-namespace x-server
```

Related Documentation

- Per Namespace Access Log Profiles on page 350
- Configuring an Access Log Profile on page 352
- Modifying the Default Access Log Profile on page 354
- Per Namespace Access Log Profiles on page 350
- Example: Access Log Profile Configuration on page 356

Logging the Origin Server IP Address in the Access Log

Using the %S (Origin Server Name) format option, you can configure Media Flow Controller to record in the access log the origin server IP address in the return response from the first four origin servers. This origin server information allows you to observe origin server utilization.

To configure an access log to log the origin server IP address:

1. Enter CLI configure mode.

   > **enable**

   # **configure terminal**

2. Format the access log to log the origin server IP address.

   (config) # **accesslog** ***log-template-name*** **format %S**

   Media Flow Controller logs up to four origin server IP addresses. For more information about configuring and formatting access logs, see the **accesslog** CLI command in the *Media Flow Controller Command Reference*.

3. View the access log formatting.

   (config) # **show accesslog** ***log-template-name***

   ```
   Log Filename : access.log
    Max Filesize : 100 MiB
    Max files to hold : 10
    Max time duration : 0 Minutes
    Format Type: w3c-ext-custom
    Format : %S %N %v
    Auto Copy URL : -Not Configured-
    Object Size to Skip : 0
    Response Codes to Skip : None
   ```

Related Documentation

- Reading Media Flow Controller Service Logs on page 340
- Configuring Media Flow Controller Service Logs Overview on page 333
- Media Flow Controller Logging Overview on page 332

- Secure Log Export Overview on page 392
- Media Flow Controller Log Codes and Sub-Codes on page 374

Configuring Media Flow Controller System Log

The system log that records all system activity such as user logins, configuration changes, and system condition changes. It does not record service activity or errors. Media Flow Controller provides several service-specific logs, detailed in "Viewing Information Using Show Commands" on page 419.

You can specify, on a global or local level, what system data is collected and when log files are deleted, or rotated; where log files are uploaded to; the format for logs; what traps are sent to system log; and whether or not to accept log messages from remote servers.

- System Log Severity Levels and Classes on page 360
- Configuring Media Flow Controller System Logging (CLI) on page 360

System Log Severity Levels and Classes

The logging commands provide predefined log severity levels so you can refine what messages are logged, and predefined log classes that sort messages according to their origin. Logging **severity-level** options are as follows:

- **emerg**—System is unusable (requires immediate action)
- **alert**—System can become unusable (requires immediate action)
- **crit**—Critical conditions (requires immediate action)
- **err**—Error conditions (minor issue; for example, "disk went bad")
- **warning**—Warning conditions (functionality okay, but suboptimal)
- **notice** —Normal but significant conditions (default)
- **info**—Informational messages (administrator actions)
- **debug**—Debug-level messages (all messages)

Logging **class** options are as follows:

- **mgmt-core**—Management daemon (mgmtd) only
- **mgmt-back**—Other back-end components
- **mgmt-front**—Front-end components, utilities, and tests

Configuring Media Flow Controller System Logging (CLI)

See **logging** in the *Media Flow Controller CLI Command Reference* for CLI details. To configure the Media Flow Controller system log:

1. Set the minimum severity of log messages to be saved in log files on local persistent storage (regardless of source), the severity level at which user-executed CLI commands are logged, or disable local logging altogether.

   logging local *severity_level*
   logging level cli commands *severity_level*
   logging local none

2. Set or remove (with **no**) a per-class override on the global logging level for logged messages local or to a specified remote system log server. All classes without an override use the global logging level set with **logging local *severity_level*** (the default is notice). Use the **no** variant or set **none** as the severity level to disable logging from that class entirely.

   logging local override class *class* priority *severity_level*
   logging *IP_address* trap override class *class* priority *severity_level*

3. Set or remove (with **no**) a remote system log server to receive log messages. Specify the severity level of logged messages sent to all, or a specified, remote system log server.

   logging *IP_address*
   logging trap *severity_level*
   logging *IP_address* trap *severity_level*

4. Stop sending log messages to any, or a specified, remote system log server.

   logging trap none logging *IP_address* trap none

5. Allow (or disable with **no**) this system to receive log messages from another host; this is disabled by default. If enabled, only log messages matching or exceeding the minimum severity specified with **logging local *log level*** are logged, regardless of what is sent.

   logging receive

6. Set (or reset default with **no**) the format in which log messages are set. The default is **standard**.

   logging format standard
   logging format welf

7. Set (or remove with **no**) the firewall name associated with each message logged in Web trends Enhanced Log Format (WELF). If no firewall name is set, the hostname is used by default. Neither command enables WELF logging if it is not already enabled with **logging format welf**.

   logging format welf fw-name *firewall_name*

8. Include (or disable with **no**) an additional field in each log message showing the number of seconds since the Epoch; the default is **disabled**. This is independent of the standard system log timestamp at the beginning of each message in the format Feb 25 18:00:00. This provides year and subsecond precision. Control the precision with the two **digits** commands that control the number of digits to the right (**fractional**) and left (**whole**) of the decimal point; **all** specifies no limit. Except for the year, all of these digits are redundant with system log timestamp.

   logging fields seconds enable

logging fields seconds whole-digits (1 | 6 | all)
logging fields seconds fractional-digits (1 | 2 | 3 | 6)

9. Configure when log files on local persistent storage should be automatically rotated. Rotation is based on time, or active log file size. The default is **daily** (once a day at midnight).

   logging files rotation criteria frequency (daily | weekly | monthly)
   logging files rotation criteria size *log_file_size_threshold*
   logging files rotation criteria size-pct *log_file_size_percent_threshold*

10. Configure how many old log files are kept. When the number of log files exceeds this number (either at rotation time, or when this setting is lowered), the system deletes as many log files as necessary to maintain this number, starting with the oldest file.

    logging files rotation max-num *maximum_number_of_files_to_keep*

11. Force an immediate rotation of the all log files.

    logging files rotation force

12. Delete a specified number of the oldest log files.

    logging files delete oldest [*number of files to delete*]

13. Upload a log file to a remote host. The **current** option specifies the current log file. To specify an archived log file, specify its number instead, as displayed by the **show log files** command.

    logging files upload (current | *file_number*) *URL*

14. View a local log file. If ***file_number*** is specified, view an archived log file, where the number is from **1** up to the number of archived log files (10 is the default maximum allowed); the higher the number, the more recent the log file. If **[not] matching *regex*** is specified, the file is filtered to only include lines either matching, or not matching, the provided regular expression. Enclose all **regex** entries in double quotes.

    show log [files *file_number*] [[not] matching *regex*]

15. Display the last few lines of the current log file, and then continue to display new lines as they come in, until you press Ctrl+C. If **[not] matching *regex*** is specified, only log lines matching, or not matching, the provided regular expression are printed. Enclose all **regex** entries in double quotes.

    show log continuous [[not] matching *regex*]

16. Display logging configuration settings or a list of local log files.

    show logging
    show log files

Example:

```
MFC (config) # logging local notice
MFC (config) # logging level cli commands notice
MFC (config) # logging 123.54.10.12
MFC (config) # logging 123.54.10.12 trap emerg
MFC (config) # logging local override class mgmt-front priority info
MFC (config) # logging 123.54.10.12 trap override class mgmt-front priority info
MFC (config) # logging 123.54.10.12 trap none
```

```
MFC (config) # logging receive
MFC (config) # logging format standard
MFC (config) # logging fields seconds enable
MFC (config) # logging files rotation criteria frequency daily
MFC (config) # logging files rotation max-num 12
MFC (config) # logging files rotation force
MFC (config) # logging files delete oldest 5
MFC (config) # logging files upload current scp://joe@sv01/home/joe
Password: *******
MFC (config) #
```

Reading the Media Flow Controller System Log

Media Flow Controller uses a system log utility for tracking and logging all types of system messages, from the merely informational to the extremely critical. This log is not specific to service activity.

Set counter thresholds with the **stats** command, and set e-mail notifications with the **email** commands. See "System Log Severity Levels and Classes" on page 360 for severity level options.

See "Configuring Media Flow Controller System Log" on page 360 for implementation details, including how to upload the log. See **logging** in the *Media Flow Controller CLI Command Reference* for CLI details.

The system log keeps track of all system activities, similar to the system log on a UNIX system. The system log provides the following information, in the following order:

- Date and time
- Media Flow Controller hostname and process name (ends with a colon)
- Process date and time, in brackets (can be missing)
- Event information, can include the severity level, in brackets; see "System Log Severity Levels and Classes" on page 360 for levels

Example:

```
Jan  6 00:10:00 MFC httpd: [Wed Jan 06 00:10:00 2010] [notice] Apache configured
 -- resuming normal operations
Jan  6 05:58:50 MFC pm[4617]: [pm.NOTICE]: Output from nknlogd (Nokeena Log
Manager): Debuglog socket 17 closed
Jan  6 08:58:27 MFC login: ROOT LOGIN ON ttyS0: user admin (System Administrator)
```

Related Documentation

- Media Flow Controller Logging Overview on page 332
- Reading Media Flow Controller Service Logs on page 340
- Secure Log Export Overview on page 392
- Media Flow Controller Log Codes and Sub-Codes on page 374

Reading the Media Flow Controller Tech Support Log

The **tech-support** command outputs a superset of system log information in the form of a compressed (.tgz) file. No configuration is necessary. The contents of the **nkn_tech-support.tgz** file include the following:

active.db
active.txt
build_version.sh
build_version.txt
config
cpuinfo.log
dmesg
iomem.log
ioports.log
list-var_opt_tms.txt
lsof.txt
lspci-vvv.log
lspci-vvvn.log
messages
messages.10.gz
messages.1.gz
messages.2.gz
messages.3.gz
messages.4.gz
messages.5.gz
messages.6.gz
messages.7.gz
messages.8.gz
messages.9.gz
mfdb
mfdb.txt
modprobe.d
nkncnt.log.1
nkncnt.log.2
output
root_cli_history
running-config.txt
scsi.log
sysinfo.txt
systemlog
version.log

To view the Media Flow Controller tech-support output, run **tech-support scp:*//URL***. You can then access the tech-support output at the specified system, move it (if needed), and view its contents.

The file is created with the name **nkn_tech-support**. We highly recommend that if you have multiple Media Flow Controllers, each Media Flow Controller be configured with a different path variable (use a different directory) so that **tech-support** data from one Media Flow Controller does not overwrite data from another. If different hosts are used for each Media Flow Controller, then the same pathname can be used.

The system logs an error in the system log if the origin server is unavailable or unreachable during a content ingest manager crawl. If an ongoing instance of the crawler has not stopped before the next refresh interval, a message is logged in the system log, as well as in the error log. Finally, a message is logged in the system log if the crawler service is down or restarted.

Example:

- System log—If the origin server is unavailable or unreachable during a content ingest manager crawl

```
syslog: Jun 13 06:08:29 vxa22-6 crawler[5828]: Crawl pcrl1 failed: Invalid
response code 504 from origin
```

- System log—If an ongoing instance of the crawler has not stopped before the next refresh interval

```
syslog:  Jun 13 01:35:00 vxa22-6 crawler[22319]: [pcrl11] Crawl interval
reached,crawl still in progress
```

Related Documentation

- Media Flow Controller Logging Overview on page 332
- Reading Media Flow Controller Service Logs on page 340
- Secure Log Export Overview on page 392
- Media Flow Controller Log Codes and Sub-Codes on page 374

Configuring Media Flow Controller Log Statistics Thresholds (CLI)

Configuring Media Flow Controller Log Statistics Thresholds (CLI)

Use the **stats** commands to set thresholds for system log events, and export parameters statistics export. See **stats** in the *Media Flow Controller CLI Command Reference* for CLI details. To configure thresholds for system log events:

1. Enable or disable (with **no**) the specified alarm. Three alarms are disabled by default: **disk_io** (disk I/O per second too high), **intf_util** (network utilization too high), and **memory_pct_used** (too much memory in use). All other alarms are enabled by default.

   stats alarm *alarm_ID* enable

   For example:

```
MFC (config) # no stats alarm total_byte_rate enable
MFC (config) # stats alarm total_byte_rate enable
```

2. Change the thresholds that initiate or terminate (clear) the specified alarm. The units for the **cpu_util_indiv** alarm are hundredths of a point of the one-minute load average. For example, setting the threshold to **100** causes an alarm if the one-minute load average is ever over 1.0 when it is sampled. The units for the **paging** alarm are number

of pages read from or written to the swap partition that has occurred over the past 20 seconds. Enter **show stats alarm** to view alarm IDs and current status. To see default threshold values, enter **show stats alarm** ***alarm_ID***. Alarm threshold defaults have been adjusted for Media Flow Controller.

```
stats alarm alarm_ID rising error-threshold threshold
stats alarm alarm_ID rising clear-threshold threshold
stats alarm alarm_ID falling error-threshold threshold
stats alarm alarm_ID falling clear-threshold threshold
```

3. Set or reset the alarm event rate-limit maximum counts for the three types of counts (**short, medium, long**) for alarms; the defaults are 5 for short, 20 for medium, and 50 for long.

```
stats alarm alarm_ID rate-limit count long (50 | duration)
stats alarm alarm_ID  rate-limit count medium (20 | duration)
stats alarm alarm_ID rate-limit count short (5 | duration)
stats alarm alarm_ID  rate-limit reset
```

4. Configure or reset alarm event rate-limit duration windows for the three types of durations for alarms; the defaults are 3600 for short, 86400 for medium, and 604800 for long.

```
stats alarm alarm_ID rate-limit window long (604800 | duration)
stats alarm alarm_ID rate-limit window medium (86400 | duration)
stats alarm alarm_ID  rate-limit window short (3600 | duration)
```

5. Change the amount of time between samples for the specified group of sample data. Enter **show stats sample** to view sample IDs and the current sampling interval.

```
stats sample sample_ID interval poll_interval_time_in_seconds
```

6. Clear all memory of the specified sample, computed historical datapoint (CHD), or alarm, or all of these together. Clearing a sample or CHD deletes all gathered data. Clearing an alarm resets it to a non-error state, clears the watermarks, and deletes the event history. Enter **show stats chd** to view CHD IDs, status, and default threshold values.

```
stats clear-all
stats sample sample_ID clear
stats chd CHD_ID clear
stats alarm alarm_ID clear
```

7. Export statistics to comma-separated value (**csv**) file, the only format supported for Version. The dataset to be exported is determined by the specified report name. If a filename is specified, the statistics are exported to a file of that name; otherwise a name is generated that contains the report name, time, and date of the export. One, both, or neither of the **after** and **before** parameters can be specified, placing boundaries on the timestamps of the instances to be exported, and they can occur in either order relative to each other. Enter **show files stats** to see the file. Enter **file stats** ***filename*** **upload** ***URL*** to send the file to another system. See also "Stats Reports Names Options" on page 367.

   stats export csv *report_name* [filename *filename*] [after *date time*] [before *date time*]

   Example:

   MFC (config) # stats export csv cpu_util
   MFC (config) # show files stats


```
cpu_util-20090401-161206.csv
MFC (config) # file stats upload cpu_util-20090401-161206.csv scp://
  joe@sv01/home/joe
Password: *******
MFC (config) #
```

8. List all statistics of the specified type or a particular stat. If **cpu** is specified, basic statistics about CPU utilization are displayed.

   show stats (alarm [*alarm_ID*] | chd [*CHD_ID*] | cpu | sample [*sample_ID*])

NOTE: You cannot set log statistics thresholds using the Web interface in Media Flow Controller Version.

Stats Reports Names Options

The following are the report names you can use to export statistics; the reports list all related alarms, chds, and samples. See **stats** in the *Media Flow Controller CLI Command Reference* for CLI details.

NOTE: All parameters you enable for statistics sampling are not exportable. The sampling can be enabled and used to generate alerts. The parameters listed under the **stats export csv ?** for CLI context help are the only parameters available for exporting. The following stats paramaters are available for exporting as csv:

```
cpu_util (CPU utilization)
memory (Memory utilization)
paging (Paging I/O)
bandwidth_day_avg (Avg. Bandwidth Usage)
bandwidth_day_peak (Peak. Bandwidth Usage)
connection_day_avg (Avg Connection Count)
connection_day_peak (Peak Connection Count)
bandwidth_week_avg (Avg. Bandwidth Usage weekly)
bandwidth_week_peak (Weekly Peak. Bandwidth Usage)
connection_week_avg (Weekly Avg Connection Count)
connection_week_peak (Weekly Peak Connection Count)
bandwidth_month_avg (Monthly Avg. Bandwidth Usage)
bandwidth_month_peak (Monthly Peak. Bandwidth Usage)
connection_month_avg (Monthly Avg Connection Count)
connection_month_peak (Monthly Peak Connection Count)
```

- **bandwidth_day_avg**—Average bandwidth usage
- **bandwidth_day_peak**—Peak bandwidth usage
- **connection_day_avg**—Average connection count
- **connection_day_peak**—Peak connection count
- **cpu_util**—CPU utilization

- **memory**—Memory utilization
- **paging**—Paging I/O (input/output)

Measurement Counters (Stats Samples)

These **stats samples** collect information used by stats alarms. The options for stats **sample ID** are shown in Table 38 on page 368; sampling interval defaults are shown in **bold**. For CLI details, see **stats** in the *Media Flow Controller CLI Command Reference* for details.

Table 38: Stats Samples

Stat Sample	Description
cache_byte_count	Bandwidth being served from RAM/buffer cache; the default is **10 seconds**.
cache_byte_count_day	Bandwidth served from RAM/buffer cache for the last 24 hours; the default is **5 minutes**.
cache_byte_count_week	Bandwidth served from RAM/buffer cache for the last 7 days; the default is **30 minutes**.
connection_count	Total active connections; the default is **10 seconds**.
cpu_util	CPU utilization: milliseconds of time spent; the default is **15 seconds**.
disk_byte_count	Bandwidth being served from disk; the default is **10 seconds**.
disk_byte_count_day	Bandwidth being served from disk for the last 24 hours; the default is **5 minutes**.
disk_byte_count_week	Bandwidth being served from disk for the last 7 days; the default is **30 minutes**.
disk_io	Disk I/O (input/output, in kilobytes); the default is **15 seconds**.
fs_mnt_bytes	File system usage, in bytes; the default is **1 minute**.
fs_mnt_inodes	File system usage, in inodes; the default is **1 minute**.
http_transaction_count	Number of HTTP transactions (GET requests) per second; the default is **10 seconds**.
interface	Network interface statistics; the default is **30 seconds**.
intf_day	Network interface statistics for the last 24 hours; the default is **5 minutes**.
intf_month	Network interface statistics for the last 30 days; the default is **4 hours**.
intf_util	Network interface utilization, in bytes; the default is **5 seconds**.
intf_week	Network interface statistics for the last 7 days; the default is **30 minutes**.
memory	System memory utilization, in bytes; the default is **20 seconds**.

Table 38: Stats Samples *(continued)*

Stat Sample	Description
num_of_connections	Current number of HTTP connections; the default is **10 seconds**.
origin_byte_count	Bandwidth being served from origin; the default is **10 seconds**.
origin_byte_count_day	Bandwidth served from origin for the last 24 hours; the default is **5 minutes**.
origin_byte_count_week	Bandwidth served from origin for the last 7 days; the default is **30 minutes**.
paging	Paging activity: page faults; the default is **20 seconds**.
perdiskbytes	Number of bytes being served from each disk drive; the default is **10 seconds**.
peroriginbytes	Number of bytes being served from origin; the default is **10 seconds**.
perportbytes	By-port I/O (input/output), in kilobytes per second; the default is **10 seconds**.
proc_mem	Memory used by Media Flow Controller processes; the default is **30 seconds**.
total_bytes	Total data bandwidth being served in the system; the default is **10 seconds**.
total_bytes_day	Total data bandwidth served in the last 24 hours; the default is **5 minutes**.
total_bytes_week	Total data bandwidth served in the last 7 days; the default is **30 minutes**.
total_cache_byte_count	Total data bandwidth being served from cache; the default is **10 seconds**.
total_disk_byte_count	Total data bandwidth being served from disk; the default is **10 seconds**.
total_origin_byte_count	Total data bandwidth being served from origin; the default is **10 seconds**.

Configuring Media Flow Controller Stats Alarms

This section describes the **stats** alarms you can set. These can reach you through an **email event name** (configurable).

Alarms can be in one of two states:

- **OK**—The alarm is in a normal state.
- **ERROR**—The alarm is already triggered and it is in the error state.

You can specify the **error-threshold** and **clear-threshold** levels for alarms using the **stats** commands; for example:

```
stats alarm total_byte_rate rising error-threshold 10
stats alarm total_byte_rate rising clear-threshold 1
```

In this example, when the **total_byte_rate** stat alarm exceeds **10**, the alarm state changes to ERROR. The state changes to OK only when the **total_byte_rate** stat alarm is` less than or equal to **1**.

Use the **stats** commands to set **error** and **clear** thresholds for **alarms**. Table 39 on page 371 shows Media Flow Controller **stats alarms**; all defaults are configurable. For details on **stats chds** (computed historic datapoints) and **stats samples**, see **stats** in the *Media Flow Controller CLI Command Reference* for details.

For configuration details, see "Configuring Media Flow Controller Log Statistics Thresholds (CLI)" on page 365.

To configure stats alarms:

1. View the current alarm, chd, and sample defaults.

   show stats (alarm | chd | sample)

2. View the alarm thresholds, chd range and interval values, and sampling interval defaults.

   show stats [alarm *alarm_ID*] [chd *chd_ID*] [sample *sample_ID*]

3. Enable a stat alarm that is disabled by default.

   stats alarm *alarm_ID* enable

4. Set the rising and falling error-thresholds and clear-thresholds, as appropriate. Some alarms operate on a "rising" basis: that is, the alarm is triggered when something rises above the error-threshold and is cleared when the level falls back to the clear-threshold. Other alarms operate on a "falling" basis: the alarm is triggered when something falls below the error-threshold and is cleared when the level rises back to the clear-threshold.

   stats alarm *alarm_ID* rising error-threshold *threshold*
   stats alarm *alarm_ID* falling clear-threshold *threshold*
   stats alarm *alarm_ID* falling error-threshold *threshold*
   stats alarm *alarm_ID* rising clear-threshold *threshold*

5. Export statistics to comma-separated value (csv) file, the only format supported for Version. The dataset to be exported is determined by the specified report name. If a filename is specified, the stats are exported to a file of that name; otherwise a name is generated that contains the report name, time, and date of the export. You can specify one, both, or neither of the **after** and **before** parameters, placing boundaries on the timestamps of the instances to be exported; they can come in either order relative to each other. Enter **show files stats** to see the file. Enter **file stats *filename* upload *URL*** to upload to another system.

   stats export csv *report_name* [filename *filename*] [after *date time*] [before *date time*]

   Example:

   ```
   MFC (config) # stats export csv cpu_util
   MFC (config) # show files stats
   cpu_util-20090401-161206.csv
   MFC (config) # file stats upload cpu_util-20090401-161206.csv scp://
     joe@sv01/home/joe
   ```



```
Password: *******
MFC (config) #
```

Table 39: Media Flow Controller Stats Alarms

Statistic	Default	Description
(Note: unless otherwise noted, the default **rising error threshold** is 200000000 Bps; the **rising clear threshold** is 100000000 Bps)		
Cache Usage Alarms		
cache_byte_rate	Disable	Total data bandwidth being served from RAM/buffer cache.
Origin Usage Alarms		
origin_byte_rate	Disable	Current total data bandwidth being served from the origin.
Disk Usage Alarms		
disk_byte_rate	Disable	Current total data bandwidth being served from disk in the system.
disk_io	Disable	Disk I/O (input/output), in kilobytes per second. The default **rising error threshold** is **5120** kilobytes per second; the default **rising clear threshold** is **4608** kilobytes per second.
CPU Usage Alarms		
cpu_util_indiv	Enable	Average CPU utilization. Units for this alarm are hundredths of a point of the one-minute load average. For example, setting this to 100 causes an alarm if the one-minute load average is ever over 1.0 when it is sampled. The default rising **error threshold** is **90** percent; the rising **clear threshold** is **70** percent.
paging	Enable	Paging activity (page faults). Units for this alarm are number of pages read from, or written to, the swap partition. The alarm is on the amount of paging activity that has occurred over the past 20 seconds. The default **rising error threshold** is **2000** page faults; the **rising clear threshold** is **1000** page faults.
total_byte_rate	Enable	Total data bandwidth being served in the system; does not include management traffic on the Ethernet port.
fs_mnt	Enable	Percent free file system space. The default **falling error threshold** is **7** percent of disk space free; the **falling clear threshold** is **10** percent of disk space free.
memory_pct_used	Disable	Percent of physical memory in current in use. The default **rising error threshold** is **90** percent of physical memory used; the **rising clear threshold** is **87** percent.
Network Interface (Port) Usage Alarms		
intf_util	Disable	Network utilization (in Bps). The default **rising error threshold** is **10485760** Bps; the **rising clear threshold** is **9437184** Bps.
perportbyte_rate	Disable	By-port I/O, in kilobytes per second.

Table 39: Media Flow Controller Stats Alarms *(continued)*

Statistic	Default	Description
connection_rate	Disable	Incoming connections per second, arrived by summing up all accepted connections and dividing by system uptime. The default **rising error threshold** is **20000** per second; the **rising clear threshold** is **10000** per second.
http_transaction_rate	Disable	Number of HTTP transactions (GET requests) per second (the number of GET requests received so far divided by system uptime). The default **rising error threshold** is **40000** per second; the **rising clear threshold** is **20000** per second.

Related Documentation

- Reading Media Flow Controller Service Logs on page 340
- Configuring Media Flow Controller Service Logs Overview on page 333
- Configuring Media Flow Controller Fault Notifications (CLI) on page 372
- Media Flow Controller Logging Overview on page 332
- Secure Log Export Overview on page 392
- Media Flow Controller Log Codes and Sub-Codes on page 374

Configuring Media Flow Controller Fault Notifications (CLI)

Configure e-mail notifications so that key people receive notice of system errors and changes.

Use the **email** commands to set e-mail addresses for fault notifications of specified, predefined, events. You can also enable autosupport e-mails: when certain failures occur Juniper Networks Support automatically receives an e-mail. See **email** in the *Media Flow Controller CLI Command Reference* for CLI details, including **email event name** options and **email class** options.

To configure fault notifications with the CLI:

1. Configure who should receive e-mail event notifications. The **class** option lets you select a class of events for the specified recipients. The **event** option lets you add specified events to the **info** event class. The **detail** option only applies to process crash events; by default it is enabled.

   email notify recipient *email_address* [class ([failure] [info])] [detail]
   email notify event *event_name*

2. Set the domain. (The default is the configured **ip domain-list**), **mailhub** (SMTP server), **mailhub-port**, return address, and include or exclude (with **no**) the return host in e-mail notifications (the default is include). The **mailhub** option must be sent for notifications to work.

   email domain (*hostname* | *IP_address*)
   email mailhub (*hostname* | *IP_address*)
   email mailhub-port *port_number*
   email return-addr *username*

email return-host

3. Specify whether event e-mails are sent to Juniper Networks for certain preconfigured events; the default is enabled. Use **no email autosupport enable** to disable.

   email autosupport enable

4. Manage undeliverable e-mails. Use **cleanup max-age** to set when to permanently delete the e-mails.

   dead-letter enable [cleanup max-age *duration*]

5. Send a test e-mail to all configured e-mail notify recipients.

   email send-test

6. Verify the configurations.

   show email

   Example:

```
MFC (config) # email notify recipient bobo@example.com
MFC (config) # email notify event disk-io-high
MFC (config) # email domain example.com
MFC (config) # email mailhub mailgate1.example.com
MFC (config) # email mailhub-port 22
MFC (config) # email return-addr support@example.com
MFC (config) # email return-host
MFC (config) # email autosupport enable
MFC (config) # email dead-letter enable
MFC (config) # show email
Mail hub: mailgate1.example.com
Mail hub port: 22
Domain: example.com
Return address: support@example.com
Include hostname in return address: yes
Dead Letter settings:
Save dead.letter files: yes
Dead letter max-age: (none)
Failure events for which emails will be sent:
process-crash: A process in the system has crashed
smart-warning: Smartd warnings
unexpected-shutdown: Unexpected system shutdown
Informational events for which emails will be sent:
liveness-failure: A process in the system was detected as hung
process-exit: A process in the system unexpectedly exited
cpu-util-ok: CPU utilization has fallen back to normal levels
cpu-util-high: CPU utilization has risen too high
disk-io-ok: Disk I/O per second has fallen back to acceptable levels
disk-io-high: Disk I/O per second has risen too high
disk-space-ok: Filesystem free space is back in the normal range
disk-space-low: Filesystem free space has fallen too low
netusage-ok: Network utilization has fallen back to acceptable levels
netusage-high: Network utilization has risen too high
memusage-ok: Memory usage has fallen back to acceptable levels
memusage-high: Memory usage has risen too high
cpu-util-ave-ok: Average CPU utilization has fallen back to normal levels
cpu-util-ave-high: Average CPU utilization has risen too high
paging-ok: Paging activity has fallen back to normal levels
paging-high: Paging activity has risen too high
Email notification recipients:
bobo@example.com (all events, in detail)
```

```
Autosupport emails
Enabled: yes
Recipient:
support@example.com
Mail hub:
mail.example.com
Events to send:
process-crash: A process in the system has crashed
liveness-failure: A process in the system was detected as hung
```

Related Documentation

- Reading Media Flow Controller Service Logs on page 340
- Configuring Media Flow Controller Service Logs Overview on page 333
- Media Flow Controller Logging Overview on page 332
- Secure Log Export Overview on page 392
- Media Flow Controller Log Codes and Sub-Codes on page 374

Media Flow Controller Log Codes and Sub-Codes

This section describes the status codes and sub-codes that can appear in the access log, error log, stream log, or other messages.

Status and Error Codes

Table 40 on page 374 describes the standard status codes that might be returned in Media Flow Controller logs. See also Table 41 on page 377 for status sub-codes and Table 42 on page 385 for Media Flow Publisher error codes.

Table 40: Logging Status (%s) HTTP Codes

Code	Description
100	Continue
200	OK (success)
201	Created
206	Partial content downloaded
250	Low on storage space
300	Multiple choices
301	Moved permanently
302	Moved temporarily
303	See other
304	Not modified

Table 40: Logging Status (%s) HTTP Codes *(continued)*

Code	Description
305	Use proxy
400	(52xxx sub-codes, except as given below) Any HTTP request parser error
401	Unauthorized
402	Payment required
403	(50007, 52004 sub-codes) Namespace lookup failure
404	Not found (90001, 90002 sub-codes) When origin server is NFS and file does not exist (52016 sub-code) Any unknown, unexpected socket closure without any sub-status code
405	Method not allowed
406	Not acceptable
407	Proxy authentication required
408	Request timeout
410	Gone
411	Length required
412	Precondition failed
413	Request entity too large
414	Request-URI too large
415	Unsupported media type
416	(52005 - 52010 sub-codes) Bad request, the requested range is not satisfiable (90003 - 90006 sub-codes) When origin server is NFS and other error code
451	Parameter not understood
452	Conference not found
453	Not enough bandwidth
454	Session not found

Table 40: Logging Status (%s) HTTP Codes *(continued)*

Code	Description
455	Method not valid in this state
456	Header field not valid for resource
457	Invalid range
458	Parameter is read-only
459	Aggregate operation not allowed
460	Only aggregate operation allowed
461	Unsupported transport
462	Destination unreachable
500	Internal server error (90008 sub-code) When origin server is NFS and NFS server is not mounted
502	Bad gateway
503	(10003 sub-code) Service unavailable: Media Flow Controller internal server is busy (for example, no CPU, malloc failure, and so on) (52026 sub-code) Limited by **resource-pool** configuration
504	(50001, 50002, 50003 sub-codes) Origin server is not reachable or not available
505	RTSP version not supported
551	Option not supported

Status and Error Sub-Codes

The Media Flow Controller status sub-codes are shown in Table 41 on page 377. See Table 40 on page 374 for status codes. Successful transactions are indicated by a **0** (zero) for the status sub-code.

Table 41: Additional Logging Status Sub-Codes

Sub-Code Object Attribute and Server System Error Codes	Module Name HTTP Response Line (if any)	Comments Counters (if any)
10001	Internal storage overflow.	Media Flow Controller stores all cacheable origin server response headers in a per-object fixed-size name/value pair table; in this case, the table size has been exceeded.
10002	Internal storage overflow.	Either the name or value length of a name/value entry exceeds the system maximum.
10003	NKN_SERVER_BUSY 404 Not Found	Session admission control due to internal resource.
Buffer Manager Error Codes		
20001	Internal error.	An incorrect parameter (length, offset, and so on) was specified in a call to the buffer manager.
20002	Internal error.	Buffer manager failed to get the contents from a provider.
20003	NKN_BUF_INVALID_OFFSET 416 Requested range not satisfiable.	Either the Start or Stop offset in the byte range is beyond the total content length. glob_cm_inval_off
20004	NKN_BUF_ORIGIN_EXPIRED	Got expired data from origin.
20005	NKN_BUF_VERSION_EXPIRED	Version referred to has expired.
20006	NKN_BUF_WAIT_FAILED	Failed to do timed wait.
20007	NKN_BUF_WAIT_LOOP	Got repeat timed wait from provider.
20008	NKN_BUF_NOT_AVAILABLE	Requested buffer not available locally.
Cache Object Descriptor (COD) Manager Error Codes		
21001	Invalid COD entry specified.	None.
21002	Entry has unknown version.	None.
21003	The request for a given object could not be satisfied because that version is no longer available.	Other than a defect, this can occur if Media Flow Controller cached (and served) part of an object and then discovered that the origin modified the object when Media Flow Controller tried to retrieve the remaining portion. Subsequent requests work correctly because Media Flow Controller discards the " stale" portion in the cache.
21004	Entry is expired.	None.

Table 41: Additional Logging Status Sub-Codes *(continued)*

Sub-Code Object Attribute and Server System Error Codes	Module Name HTTP Response Line (if any)	Comments Counters (if any)
Server-Side Player Error Codes		
30001	NKN_SSP_BAD_URL	Malformed URI. Virtual player cannot decipher the URL and/or its query parameters.
30005	Connection close directive from SSP.	None.
30011	NKN_SSP_HASHCHECK_FAIL	SSP was unable to authenticate the request and closed the connection.
30021	NKN_SSP_INSUFF_QUERY_PARAM	Incoming request did not match any query parameters.
30023	NKN_SSP_MISSING_QUERY_PARAM	A required query parameter was absent in the incoming request.
30035	NKN_SSP_CHUNK_METAFILE_NOT_SUPP	Chunked meta file reads are not supported by SSP from cache.
30037	NKN_SSP_CONTAINER_FRMT_NOT_SUPP	Unsupported video container format requested for server side processing and/or delivery.
30041	NKN_SSP_BAD_QUERY_OFFSET	Invalid or unsupported value for a query parameter.
30043	NKN_SSP_PYTHON_PLUGIN_FAILURE	Call to the Python plug-in was unsuccessful.
30045	NKN_SSP_CLI_PARAM_UNSPECIFIED	CLI parameter was not specified as part of the configuration.
30050	NKN_SSP_VIDEOID_DISABLED	Video ID field for Type-5 virtual player has not been enabled; cannot cache YouTube videos.
30051	NKN _SSP_BAD_REMAPPED_URL	SSP (server-side player) remapped the request URL to an internal URL and received a 404 response from the origin server when attempting a fetch.
SmoothStream-Pub Virtual Player Error Codes		
30404	NKN_SSP_SS_NULL_BUF_INT_FETCH	An internal fetch for manifest returned an empty buffer.
30408	NKN_SSP_SS_MFRO_OFF_FAIL	The media fragment random offset (MFRO) was not found in the ISMV or ISMA files.
30400	NKN_SSP_SS_TRACK_NOT_FOUND	Unable to find a media track in the ISMV package corresponding to the bit rate and track type specified by the manifest.

Table 41: Additional Logging Status Sub-Codes *(continued)*

Sub-Code Object Attribute and Server System Error Codes	Module Name HTTP Response Line (if any)	Comments Counters (if any)
30402	NKN_SSP_SS_SVR_MANIFEST_ISM_NOT_FOUND	Server manifest file was not found as part of the SmoothStreaming package.
30406	NKN_SSP_SS_ISMV_ISMA_NOT_FOUND	The ISMV or ISMA file was not found.
30410	NKN_SSP_SS_MEM_ALLOC_HDR_FAIL	Memory allocation error.
Flashstream-Pub Virtual Player Error Codes		
30450	NKN_SSP_FS_URI_FORMAT_ERR	URI does not conform to Flashstream, the recommended format.
30452	NKN_SSP_FS_SEGFRAG_TAG_NOT_FOUND	Mandatory Seg, Frag, and or Delimiter tag is missing in request.
30454	NKN_SSP_FS_F4X_NOT_FOUND	The Flash F4X index file was not found.
30456	NKN_SSP_FS_MEM_ALLOC_HDR_FAIL	Memory allocation error.
30458	NKN_SSP_FS_F4X_CONTEXT_FAILURE	Failed to create an F4X context.
30460	NKN_SSP_FS_F4F_OFFSET_PARSE_ERROR	Failed to parse the fragment offset in the F4F file.
Initialization and Disk Management Errors		
40001	An unknown disk cache name was given.	None.
40002	An issued command is incorrect given the state of the disk.	None.
40003	During a **status active** command, the system could not find the requested device.	None.
40004	A **media-cache disk *disk_name* deactivate** command failed (most likely an un-mount failed).	None.
40005	NKN_DM2_DISK_ADMIN_ACTION	A previous command was given to cause the disk to be in the “ cache enabled” state.
40006	NKN_DM2_BITMAP_NOT_FOUND	A startup error occurred.
40007	NKN_DM2_EMPTY_CACHE_TIER	A request was sent to a disk tier that has no enabled disks.
40008	A **media-cache disk *disk_name* format** command failed.	None.

Table 41: Additional Logging Status Sub-Codes *(continued)*

Sub-Code Object Attribute and Server System Error Codes	Module Name HTTP Response Line (if any)	Comments Counters (if any)
40009	A **media-cache disk *disk_name* status active** command failed.	None.
40010	NKN_DM2_EVICTION_SKIPPED	An eviction request was made but the request had some type of error in it.
40011	NKN_DM2_OBJ_NOT_YET_WRITTEN	A startup error occurred.
40013	NKN_DM2_WRONG_CACHE_VERSION	Attempting to enable a cache that has the wrong version number. This should never happen because older versions should be converted to the current version. Media Flow Controller doesn't downgrade cache versions.
40014	Format conversion failed during a **status active** or **format** command.	None.
41001	NKN_DM2_DISK_CACHE_NOT_FOUND	An unknown disk cache name was given to the CLI.
41002	NKN_DM2_INVALID_MGMT_REQUEST	The command attempted was not valid for the given state of the disk. This is a generic error.
41003	NKN_DM2_DISK_DEVICE_NOT_FOUND	During a "status active" command, the system could not find the requested device.
41004	NKN_DM2_DISK_DEACTIVATE_FAILED	A **media-cache disk *disk* status inactive** command was issued and failed.
41005	NKN_DM2_FORMAT_FAILED	A **media-cache disk *disk* format** command was issued and failed.
41006	NKN_DM2_DISK_ACTIVATE_FAILED	A **media-cache disk *disk* status active** command was issued and failed.
41007	NKN_DM2_CONVERT_FAILED	When Media Flow Controller is starting or a drive is made "status active," a possible format conversion is done. If that conversion fails, Media Flow Controller returns this error.
41008	NKN_DM2_MUST_CACHE_DISABLE	If a drive is in the cache enabled state and a status inactive command is given, we return this error.
41009	NKN_DM2_MUST_CLEAR_ERROR	If a drive is in an error state of any kind and a cache enable command is given, Media Flow Controller returns this error.
41010	NKN_DM2_MUST_STATUS_ACTIVE	If a drive is in the status inactive state and a cache enable command is given, this error is returned.

Table 41: Additional Logging Status Sub-Codes *(continued)*

Sub-Code Object Attribute and Server System Error Codes	Module Name HTTP Response Line (if any)	Comments Counters (if any)
50001	Premature close on an origin server connection due to an inactivity timeout or an unexpected condition.	None.
50002	Internal resource allocation error.	None.
50003	Target origin server is unavailable.	None.
50005	Connection error on attempt to establish a connection to the target origin server.	None.
50006	Unexpected condition when building the HTTP request for the target origin server.	None.
50007	Unable to determine the namespace associated with the HTTP request when attempting to resolve a cache miss using the origin server provider.	None.
50008	Internal system inconsistency when attempting to resolve a cache miss using the origin server provider.	None.
50009	Internal error.	None.
50011	Parse error processing origin server response headers.	None.
50012	Parse error processing Content-Range header.	None.
50013	Parsed Content-Range data inconsistent with internal state.	None.
50015	Attempt to store query string data failed.	None.
50016	Attempt to save object validation-specific data failed.	None.
50018	Unexpected internal error when attempting to initiate an HTTP GET to the origin server.	None.
50019	Internal error between HTTP and OM.	None.
50020	Conversion of origin server response data to attribute data failed.	None.

Table 41: Additional Logging Status Sub-Codes *(continued)*

Sub-Code Object Attribute and Server System Error Codes	Module Name HTTP Response Line (if any)	Comments Counters (if any)
50021	System inconsistency detected. Data retrieved by OM on for client is inconsistent with existing data in cache.	None.
50022	Attempt to store origin server response data for chunk encoded object failed.	None.
50023	Origin manager received an HTTP 404 (Not Found) response from the origin server for a namespace with SSP configured.	None.
50024	Attempt to save object validation specific data for chunk encoded object failed.	None.
HTTP Error Codes		
52002	Internal error (low system memory).	None.
52003	Client sent malformed GET request.	None.
52004	NKN_HTTP_NAMESPACE 404 Not Found.	Failed to match any namespace in configuration. glob_http_namespace_err_52004
52005	NKN_HTTP_REQ_RANGE_1 416 Requested Range Not Satisfiable.	Either the Start or Stop byte range in the request is negative.
52006	NKN_HTTP_REQ_RANGE_2 416 Requested Range Not Satisfiable	Stop offset is less than the start offset in the byte range.
52007	Client request byte range inaccurate.	None.
52010	Origin server response byte range inaccurate.	None.
52011	Requested content type does not match content.	None.
52012	NKN_HTTP_URI 400 Bad Request.	URI does not exist in the HTTP request.
52014	URI length must be between 1 and 256 bytes.	URI_MORE means the requested URI exceeded 256 bytes; URI_LESS means it was less than 1 byte.
52015	Internal error.	Problem with seek function.

Table 41: Additional Logging Status Sub-Codes *(continued)*

Sub-Code Object Attribute and Server System Error Codes	Module Name HTTP Response Line (if any)	Comments Counters (if any)
52016	NKN_HTTP_CLOSE_CONN 404 Not Found.	Failed to tunnel the request because origin server is NFS or server map. glob_http_close_conn_err_52016
52018	NKN_HTTP_NO_HOST_HTTP11 400 Bad Request.	HTTP/1.1 request without Host field.
52019	NKN_HTTP_PARSER 400 Bad Request.	Malformed HTTP request; Media Flow Controller cannot parse the request. glob_http_parse_err_nulls glob_http_parse_err_req_rate glob_http_parse_err_uri_tolong glob_http_parse_err_req_toshort glob_http_parse_err_normalize_uri glob_http_parse_err_badreq_1 glob_http_parse_err_badreq_2 glob_http_parse_err_badreq_3 glob_http_parse_err_longcookie glob_http_parse_err_badreq_4 glob_http_parse_err_badreq_5
52020	NKN_HTTP_ERR_CHUNKED_REQ 400 Bad Request.	Chunked encoding request.
52021	NKN_HTTP_BAD_HOST_HEADER 400 Bad Request.	Domain in HTTP request is not an FQDN domain.
52022	NKN_HTTP_BAD_URL_HOST_HEADER 400 Bad Request.	Domain in HTTP absolute URI is not an FQDN domain.
54001	NKN_PE_REJECT	Request rejected because of an administrator-defined policy.
54002	NKN_PE_REDIRECT	Request redirected because of an administrator-defined policy.

Table 41: Additional Logging Status Sub-Codes *(continued)*

Sub-Code Object Attribute and Server System Error Codes	Module Name HTTP Response Line (if any)	Comments Counters (if any)
54101	NKN_PE_REJECT_ON_GEOIP_FAIL	Request rejected because of a GEO database lookup failure. (The bypass on the GEO lookup failure is not configured.)
54102	NKN_PE_GEOIP_REJECT	Request rejected because of an administrator-defined GEO policy.
54103	NKN_PE_GEOIP_REDIRECT	Request redirected because of an administrator-defined GEO policy.
Media Manager Error Codes		
70001	NKN_MM_GET_URI_ERR	Error in GET request.
70002	NKN_MM_STAT_URI_ERR	Error in STAT request.
70003	NKN_MM_PROMOTE_PUT_ERR	Error in cache write to disk.
70004	NKN_MM_PROMOTE_GET_ERR	Error in fetching data for ingest or promote.
70005	NKN_MM_PROMOTE_ATTR_ERR	Error in fetching attributes for ingest or promotion.
70006	NKN_MM_PROMOTE_GEN_ERR	Generic error during preparation for ingest or promotion.
70007	NKN_MM_COD_ERR	COD open failure during a GET request.
70008	NKN_MM_COD_ERR	COD open failure during a STAT request.
70009	NKN_MM_OBJ_NOT_CACHE_WORTHY	Object not worthy to be ingested into cache.
70010	NKN_MM_TOKEN_ERR	Namespace error during ingest or promotion.
70020	NKN_MM_CONF_RETRY_REQUEST	Retry request from origin manager.
70021	NKN_MM_UNCOND_RETRY_REQUEST	Unconditional retry request from origin manager.
70022	NKN_MM_OBJ_NOT_FOUND_IN_CACHE	Object not found in cache.
71002	Internal error: memory allocation error.	None.
71003	Internal network API error.	None.
71005	Error in the network socket.	None.
71006	Error in network connection.	None.

Table 41: Additional Logging Status Sub-Codes *(continued)*

Sub-Code Object Attribute and Server System Error Codes	Module Name HTTP Response Line (if any)	Comments Counters (if any)
NFS Origin Manager Error Codes		
90001	NKN_NFS_STAT_URI_ERR	Error during a STAT request to NFS origin.
90002	NKN_NFS_GET_URI_ERR	Error during a GET request to NFS origin.
90003	NKN_NFS_STAT_INV_OFFSET	Invalid offset in a STAT or GET request to origin.
90005	NKN_NFS_GET_SERVER_BUSY	Too many GET requests queued for the NFS origin.
90007	NKN_NFS_VERSION_MISMATCH	Object version mismatch occurred during a GET request.
90008	NKN_NFS_MOUNT_ERR	Error during NFS mount.

Table 42: Media Flow Publisher Log, Error Codes

Error Code Generic Media Flow Publisher Errors	Module	Description
-1	E_MFP_NO_MEM	Ran out of memory.
-2	E_MFP_INVALID_FILE	Invalid input or source files.
-3	E_MFP_INVALID_SESS	Invalid session ID or session ID collision.
-4	E_MFP_INVALID_CONVERSION_REQ	Unsupported conversion request (FILE conversion only).
-5	E_MFP_SOCK_CREATE	Unable to create a socket fd.
-6	E_MFP_BIND_FAIL	Unable to bind to port specified in PMF file.
-7	E_MFP_NO_SPACE	Unable to bind to port specified in PMF file.
PMF Parsing Errors		
-1000	E_MFP_PMF_INVALID_FILE	Invalid PMF file.
-1001	E_MFP_PMF_NO_MEM	Run out of memory.

Table 42: Media Flow Publisher Log, Error Codes *(continued)*

Error Code Generic Media Flow Publisher Errors	Module	Description
-1002	E_MFP_PMF_INVALID_TAG	Invalid tag.
-1003	E_MFP_PMF_INVALID_ARG	UNUSED
-1004	E_MFP_PMF_UNSUPPORTED_TYPE	Source type is neither FILE nor LIVE.
-1005	E_MFP_PMF_INCOMPLETE_TAG	Missing child tag.
-1006	E_MFP_PMF_DOC_PTR_FAILED	libxml unable to parse the file.
-1007	E_MFP_PMF_XSD_MISMATCH	XSD compliance fails.
-1008	E_MFP_PMF_XSD_INT_ERROR	XSD parser fails.
-1009	E_MFP_PMF_INSUFF_MEDIA_TYPE	Missing parameters in the pub scheme.
-1010	E_MFP_PMF_INCOMPATIBLE_MEDIA_TYPE	Unsupported Media type tag.
TS Formatter Errors		
-3001	E_MFP_FRUIT_INIT	Initialization of APPLE formatter fails.
-3002	E_MFP_SL_INIT	Initialization of Silverlight fails.
File Conversion Errors		
-105	E_VPE_FILE_CONVERT_CONTINUE	Continue file conversion.
-106	E_VPE_FILE_CONVERT_STOP	Stop file conversion.
-107	E_VPE_FILE_CONVERT_SKIP	Skip file conversion.
-108	E_VPE_FILE_CONVERT_CLEANUP_PENDING	Cleanup pending for file conversion.
-109	E_VPE_INVALID_FILE_FROMAT	Invalid file format.
Generic Parser Error Codes		
-201	E_VPE_PARSER_NO_MEM	Out of memory.
-202	E_VPE_PARSER_INVALID_FILE	Unable to open file descriptor.
-203	E_VPE_PARSER_INVALID_OUTPUT_PATH	Invalid output path (unable to write file).
-204	E_VPE_PARSER_FATAL_ERR	Fatal error, invalid state change.

Table 42: Media Flow Publisher Log, Error Codes *(continued)*

Error Code Generic Media Flow Publisher Errors	Module	Description
-205	E_VPE_PARSER_INVALID_CTX	Invalid context or state.
-206	E_VPE_MAX_PROFILE_ERR	The num:of: profiles exceed the MAXIMUM assigned.
MP4 Parsing Error Codes		
-301	E_VPE_MP4_MOOV_ERR	Error in parsing MOOV box.
-302	E_VPE_MP4_NO_STSS_TAB	No STSS box.
-303	E_VPE_NO_AVCC_BOX	No AVCC box.
-304	E_VPE_MP4_MISALIGMENT	Syncpoints across MP$ profiles are misaligned.
-305	E_VPE_MP4_ESDS_ERR	Error in parsing ESDS box.
-306	E_VPE_MP4_AV_SYNC_ERROR	Audio and video sync points are not equal.
-307	E_VPE_PROFILE_LEVEL_ERROR	Profiles other than baseline 3.0 will display this error.
-308	E_VPE_VIDEO_CODEC_ERR	Error in parsing video codec data.
-309	E_VPE_DIFF_MOOFS_ACROSS_PROFILE	The number of moofs is different across the profiles.
ISMx Parsing Related Error Codes		
-401	E_ISMV_INVALID_CTX	Invalid context.
-402	E_ISMV_NOMEM	Out of memory.
-403	E_ISMV_INVALID_TIMESCALE	Invalid track timescale.
-404	E_ISMV_NO_MORE_FRAG	No more fragments to parse.
-405	E_ISMC_PARSE_ERR	Required tag missing in ISMC file.
-406	E_ISM_PARSE_ERR	Required tag missing in ISM file.
-407	E_ISMV_PARSE_ERR	Required tag missing in ISMV file.
-408	E_ISMV_MFRA_MISSING	MFRA is missing.
HTTP Sync Reader Error Codes		
-450	E_VPE_HTTP_SYNC_VER_ERR	Invalid HTTP version number.

Table 42: Media Flow Publisher Log, Error Codes *(continued)*

Error Code Generic Media Flow Publisher Errors	Module	Description
-451	E_VPE_HTTP_SYNC_PARSE_ERR	HTTP parser error.
F4V Parser Error Codes		
-501	E_VPE_F4V_INVALID_F4M	Invalid F4m.
-502	E_VPE_F4V_INVALID_FRAG	Invalid fragment.
-700	E_VPE_VID_ONLY_NOT_SUPPORTED	Only video is present.
-701	E_VPE_AUD_ONLY_NOT_SUPPORTED	Only audio is present.
TS Creation Error Codes		
-800	E_VPE_TS_CREATION_FAILED	Failed to create TS.
-801	E_VPE_NO_VIDEO_PKT	No video packets.

Access Log Reason Codes and Descriptions

Table 43 on page 388 displays the access log (%c) reason codes and descriptions for each tunneled activity.

Table 43: Access Log Reason Codes and Descriptions for Tunnel Activity

Code	Description
Reason Code (1-63) for Taking Tunnel in Request Code Path	
1	A bad request from the client.
2	The origin selection method is absolute-url, but the request is not cacheable.
3	There is no "Host" header.
4	The configuration is set to non-cacheable.
5	The configured virtual-player is set to force tunnel.
6	The request for Cached Object Descriptor (COD) returned NULL.
7	The URI size is greater than 512 bytes (it exceeds the URI size limit).
8	The request is neither GET nor HEAD.

Table 43: Access Log Reason Codes and Descriptions for Tunnel Activity *(continued)*

Code	Description
9	A response with an unknown content-type; the transaction is tunneled. This is only valid when Media Flow Controller is configured to handle content-type. In most cases, it is irrelevant.
10	Force tunnel for all requests.
11	A response has a “Content-Length: 0” header. The response is tunneled.
12	A multibyte MIME header exists in the request.
13	Some data exists in addition to HTTP headers.
14	An authorization header is present.
15	A cookie header is present.
16	The request has query parameters.
17	The request has a “Cache-Control” header set to no-cache.
18	The request has a “Pragma” header set to no-cache.
19	The request has transfer encoding chunked.
20	The host exists but it is not a valid fully qualified domain name (FQDN).
21	The host in the absolute URI is not a valid FQDN.
22	The URI does not exist in the request.
23	The byte-range request has a negative value.
24	The HTTP version is not 1.0 or 1.1.
25	HTTP version 1.1 is without a “Host” header.
26	A transfer type encoding request.
27	Byte-range request: Stop range is less than start range. For example, in a request with “Byte-range: bytes 100-3,” the start data offset is larger than stop data offset.
28	The dynamic URI regex match failed.
29	The request is tunneled because the HTTP request header size is larger than the default 8K bytes. We can configure to parse up to 32K bytes.
30	The request encoding type is not supported.

Table 43: Access Log Reason Codes and Descriptions for Tunnel Activity *(continued)*

Code	Description
Reason Code (65-127) for Taking Tunnel in Response Code Path	
65	Policy Engine is set to do not cache.
66	Received URI with query string and query-string action no-cache is configured.
67	Both transfer-encoding: byte-range and transfer_encoding: chunked exist.
68	Neither the content_length or transfer_encoding: chunked exist.
69	The content_length value is zero.
70	The URI has unprintable characters, and virtual-player is not configured.
71	Received a 404 from the server, and virtual-player is not configured.
72	Received an unsupported response from the server.
73	Tunneling a 302 response in transparent proxy mode.
74	Received a 302 response without a "Location" header.
75	Received a 302 response and the "Location" header has errors.
76	The COD verification failed.
77	A response with "Cache-Control: no-transform" header. The response is tunneled.
78	Failed to save request state data for validation.
79	Failed to save MIME header.
80	Failed to serialize mime_header_t into attribute buffer. Internal error. Failed to convert the HTTP response header into the correct internal mime structure.
81	Error in getting content-range from response data.
82	The response has invalid content-range data.
83	No content length data.
84	Received origin-server data-offset that is not the same as the configured request-offset. For example, when a request offset range is 10 to 200 bytes, and the origin server returns offset 300 to 400 bytes.
85	The response has "Cache-Control" header set to private.
86	The response has the "Set-Cookie: cooke_name=cookie_value" header. The response is tunneled.

Table 43: Access Log Reason Codes and Descriptions for Tunnel Activity *(continued)*

Code	Description
87	The object is already expired; caching is disabled.
88	Adding a query to attributes failed.
89	Cache Object Size. In Media Flow Controller Release 2.0.7, you can cache a response with a boundary. For example, you can configure to only cache a response with a content-length between 1 MB to 100 MB and tunnel any response body beyond this range.
90	Non-cacheable, because of some cache-control directive.
91	The client request size is more than the configured one.
92	A chunked response, but cache-age is less than what is configured.
93	The response encoding type is not supported.
94	The cache index-based response has no configured header value in reply.
95	The cache index-based response URI exceeds the maximum URI size
96	The cache index-based response has out-of-range data bytes.
97	A dynamic URI tokenization error.
98	The cache index response data is configured for a chunked object.

NOTE: **Reason codes greater than 100000 are for internal generated tunnel requests.**

Related Documentation

- Secure Log Export Overview on page 392
- Configuring Media Flow Controller Service Logs (CLI) on page 338
- Media Flow Controller Logging Overview on page 332
- Configuring Media Flow Controller Service Logs Overview on page 333

Secure Log Export Overview

Media Flow Controller supports both push and pull log file generation. You can configure only one type of log generation can at a time. In push log generation:

- Logs are uploaded by Media Flow Controller to an external, configured server.
- Uploads occur when a log file is closed or rotated.
- Protocols, such as SCP, SFTP, FTP are supported.

In pull log generation, Media Flow Controller exports certain log files to a log export folder from which an only an authorized user can access and download the log files to an external log collector using the SSH File Transport Protocol (SFTP). The log export folder is secure and isolated in Media Flow Controller so there is no hazard of downloading any other files. (In the current release, crawler logs and access logs are supported).

Secure log pull from Media Flow Controller to a log collector provides the following benefits:

- Simplifies log file management on Media Flow Controller by centralizing the responsibility in a central log management server
- Enables a log management server to pull logs from Media Flow Controller using SFTP
- Log management server has full control over handling log transfer failures and error recovery
- Log transfers can occur during quiet periods to optimize the network usage

To configure secure log pull, you must:

- Enable the Media Flow Controller SSH service. SFTP listens on the same port as the SSH service. For information about configuring SSH, see "Configuring RADIUS, TACACS+, LDAP, and SSH (Web Interface)" on page 300. If SSH is enabled, the SFTP service is also enabled.
- Enable a LogTransferUser account for logiin access to the Media Flow Controller log export location using the **[no] username LogTransferUser disable** CLI command. The LogTransferUser account is disabled by default.
- Configure the properties for how the Media Flow Controller logs are purged from the log export location, at periodic intervals or at storage size thresholds, using the **logging export purge** CLI command. The default purge frequency is 6 hours. The default purge size age is 85 percent.

Related Documentation

- Configuring the Secure File Transport Protocol for Log Export on page 393
- Configuring the LogTransferUser Account for Secure Log Export Using SFTP on page 393
- Restricting IP Hosts or Subnets to Which the LogTransferUser Can Log on page 395
- Logging In to Media Flow Controller as LogTransferUser for Secure Log Export on page 396

- Purging Secure Log Transfer Files on page 397

Configuring the Secure File Transport Protocol for Log Export

Media Flow Controller exports certain log files to a log export directory and allows a privileged user—the LogTransferUser—to access those log files using SFTP for downloading to an external log collector. SFTP listens on the same port as the one configured for the SSH service. For the SFTP service to work, you must enable the SSH encryption protocol for file transfers.

FTP, currently running on port 21, is disabled by default for security reasons. FTP is not secure and uses clear-text passwords for authentication. If FTP is enabled, use the **service stop mod-ftp** CLI command to disable it. If FTP is disabled, you can enable it usng the **service restart mod-ftp** CLI command.

To verify that SFTP is enabled for log access and download:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Ensure that SSH is configured.

   (config) # **show ssh server**

   For more information about configuring SSH, see"Configuring RADIUS, TACACS+, LDAP, and SSH (Web Interface)" on page 300 and the **ssh server** command in the *Media Flow Controller CLI Command Reference*.

Related Documentation

- Secure Log Export Overview on page 392
- Restricting IP Hosts or Subnets to Which the LogTransferUser Can Log on page 395
- Configuring the LogTransferUser Account for Secure Log Export Using SFTP on page 393
- Logging In to Media Flow Controller as LogTransferUser for Secure Log Export on page 396
- Purging Secure Log Transfer Files on page 397

Configuring the LogTransferUser Account for Secure Log Export Using SFTP

Media Flow Controller creates a default user account—LogTransferUser—during system manufacture or upgrade. The LogTransferUser accesses certain log files from a Media Flow Controller and downloads them to an external log collector.

The LogTransferUser belongs to the LogTransferUser group that has privileges to log in to Media Flow Controller using the SSH File Transport Protocol (SFTP) with access to only the log export directory and restricted read access to transfer logs.

The LogTransferUser account is disabled by default. You must enable it.

This topic includes the following tasks:

- Enabling the Default LogTransferUser Account on page 394

Enabling the Default LogTransferUser Account

To enable the default LogTransferUser:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Enable the LogTransferUser account.

   (config) # **no username LogTransferUser disable**

   Only the Media Flow Controller administrator is allowed to enable or disable the LogTransferUser account. You cannot remove this default user from the system. The default LogTransferUser account includes:

   - Account Enabled: no
   - Home Directory: /home/LogExport
   - Password: <NOT Configured>
   - Capability: LogTransferGroup

   To disable the LogTransferUser account, use the **username LogTransferUser disable** CLI command.

3. View the default LogTransferUser account details.

   (config) # **show usernames**

   If the LogTransferUser account is disabled:

   ```
   USERNAME        FULL NAME         CAPABILITY  ACCOUNT STATUS
   LogTransferUser LogTransferUser   LogTransferGroup Local password login disabled
   ```

   If the LogTransferUser account is enabled:

   ```
   cmbu-vxa20-15 (config) # username LogTransferUser password 0 foobar
   cmbu-vxa20-15 (config) # show usernames
   USERNAME              FULL NAME          CAPABILITY  ACCOUNT STATUS
   LogTransferUser LogTransferUser          LogTransferGroup          Password set
   ```

Related Documentation

- Secure Log Export Overview on page 392
- Configuring the Secure File Transport Protocol for Log Export on page 393
- Restricting IP Hosts or Subnets to Which the LogTransferUser Can Log on page 395
- Logging In to Media Flow Controller as LogTransferUser for Secure Log Export on page 396

- Purging Secure Log Transfer Files on page 397

Restricting IP Hosts or Subnets to Which the LogTransferUser Can Log

The **ip filter** command restricts the IP hosts or subnets from which the LogTransferUser can log in to Media Flow Controller to download log files.

NOTE: **The ip filter command is a global configuration, and is not specific to the LogTransferUser user account.**

To restrict IP hosts and subnets:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Enable IP filtering.

   (config) # **ip filter enable**

   For more detailed information about this global configuration, see the **ip** command in the *Media Flow Controller CLI Command Reference*.

3. Display the active IP filtering configuration.

   (config) # **show ip filter**

   This command lists network filtering rules currently installed on the system, omitting those that are not configured on this system.

   NOTE: **There might be a reduction in the system media delivery performance when you enable ip filter is enabled.**

Related Documentation

- Secure Log Export Overview on page 392
- Configuring the Secure File Transport Protocol for Log Export on page 393
- Configuring the LogTransferUser Account for Secure Log Export Using SFTP on page 393
- Logging In to Media Flow Controller as LogTransferUser for Secure Log Export on page 396
- Purging Secure Log Transfer Files on page 397

Logging In to Media Flow Controller as LogTransferUser for Secure Log Export

The LogTransferUser can log in to Media Flow Controller using the SSH File Transport Protocol (SFTP) to a centralized log file location. There is no limit on the number of sessions or connections that a LogTransferUser can open to Media Flow Controller.

NOTE: There is no restriction in allowing a valid user to connect to the SSH service; however, when a user supplies incorrect login credentials, the session is terminated.

The LogTransferUser has read-only access within the log export directory. The LogTransferUser can download log files that are completed or closed by the logger service.

To log in to Media Flow Controller as the default LogTransferUser:

1. Using an SFTP client, log in to the Media Flow Controller management IP address.
2. Enter your default username.

   LogTransferUser

   # sftp **LogTransferUser@10.157.43.84**

   The default LogTransferUser has no password.

   ```
   Juniper Networks Media Flow Controller

   Password:
   sftp> pwd
   Remote working directory: /home/LogExport
   sftp>
   ```

3. If you have a password, enter it.

   The Media Flow Controller **/home/LogExport** directory appears.

   # **sftp>**

   If login fails, the administrator should perform the following:

   - Check the system log to determine the cause of login failures using the **show log command**.
   - Check whether the LogTransferUser is enabled using the **show username** CLI command.
   - Check whether the Media Flow Controller, SSH server is listening on port 22 on the interface IP address on which the user is trying to log in.

4. Copy the log files from Media Flow Controller.

Only log file copying is allowed. The LogTransferUser is not allowed to delete, move, or rename files. File copying is done using SFTP commands, such as **get**. Refer to the specific SFTP client for information on how to copy files.

5. Exit the secure shell.

Related Documentation

- Secure Log Export Overview on page 392
- Configuring the Secure File Transport Protocol for Log Export on page 393
- Configuring the LogTransferUser Account for Secure Log Export Using SFTP on page 393
- Restricting IP Hosts or Subnets to Which the LogTransferUser Can Log on page 395
- Purging Secure Log Transfer Files on page 397

Purging Secure Log Transfer Files

The log files are purged periodically based on:

- Frequency, which is time based: From 1 hour up to 24 hours. The default is 6 hours.
- Size, which is the percentage of storage space. The default is 85 percent of disk storage space.

The Media Flow Controller administrator can configure the purge frequency and size criteria using the **log export purge** CLI command. You can force purge all downloadable log files using the **logging export purge force** CLI command. When you enter this command, the syslog displays a list of the export log files that were forcibly purged.

Only log files that are closed are available for download from the log export directory.

Log rotation policies that are configured on a per access-log profile or crawl-log are enforced on the profile, but they are not enforced on the log export folder.

The filename formats of log files in the export directory are as follows:

- The filename pattern for access logs:

  *hostname*_accesslog_*log-template-name_version_id_start_time_end_time*.gz

  For example, mfc-101_accesslog_default_0_201205111155_201205111200.gz

  NOTE: The *version_id* is 0 by default. If more than one log file is rotated within the rotation interval between *start_time* and *end_time* as configured under the access log profile rotation interval, then *version_id* isincremented by 1. It is again reset to 0 in the next interval.

  The *start_time* and *end_time* are in the format of *YYYYMMDDHHmm*. The *end_time* is the time when the log file is closed, based on the rotation interval configuration. The time is in UTC format.

  For example, QA10_accesslog_default_1_201205220741_201205220754.gz

- The filename pattern for the crawl logs is:

  *hostname*_crawl_log_<version_id>_<endtime>.gz

  For example: QA10_crawl_log_0_20120522075418.gz

Log files in the log export directory need not be in time contiguous order, as some files may be missing due to service downtime.

A file may be partial in the log Export directory due to abrupt service outages or as governed by existing accesslog file rotation or closure policies.

The header record in the log files is subject to existing log policies:

- HTTP access log files can include a header indicating format, version, start time, and so on if the log profile is configured for such.
- Crawl logs cannot include a header format.

The trailer record isthe last line in the log file. The format is:

Last Modified : <endtime> endtime: yyyymmddhhmmss

For example: # Last Modified : 20120522111323

The trailer might be missing if the log file is not closed properly.

To define a policy to purge log files in the log export directory:

1. Enter the CLI configure mode.

   > **enable**

   # **configure terminal**

2. Configure the log file purge frequency. in hours.

   (config) # **logging export purge criteria frequency *hours***

   The default purge frequency is 6 hours.

   For a frequency-based purge, allow the configuration to be between 0 and 24 hours. Disable frequency-based deletion by setting the hours to 0.

3. Configure purging log files when they surpass a specified percentage of storage.

   (config) #**logging export purge criteria size-pct *percentage***

   For size based purge, allow the configuration to be between 0.001 and 100 percent.

4. Optionally, purge all of the log files immediately by force.

   (config) #**logging export purge force**

5. Show the export logging configuration.

   (config) # **show logging export**

   ```
   Logexport purge size threshold-pct: 85.000% of partition (34269.72 megabytes)
   Logexport purge interval: 6 hr
   ```


For example:

```
logging export purge
  criteria frequency <hours>
  criteria size-pct <percentage>
  force
```

Related Documentation

- Secure Log Export Overview on page 392
- Configuring the Secure File Transport Protocol for Log Export on page 393
- Configuring the LogTransferUser Account for Secure Log Export Using SFTP on page 393
- Restricting IP Hosts or Subnets to Which the LogTransferUser Can Log on page 395
- Logging In to Media Flow Controller as LogTransferUser for Secure Log Export on page 396

CHAPTER 13

SNMP Support

SNMP Support Overview

SNMP is a network management protocol used to monitor network-attached devices for fault conditions or problems. An SNMP-managed network consists of three key components: the managed device or devices, the “agent” (configured software on managed devices), and the network management system (NMS) software. Media Flow Controller provides an SNMP agent that can be configured to monitor various system statistics and parameters.

SNMPv1 and SNMPv2 monitoring are supported by the Media Flow Controller. The manager to agent *SetRequest* function used to change the value or configuration of a variable is not supported. SNMPv3 is not currently supported.

The Media Flow Controller's implementation of SNMP has a large number of statistical counters to monitor device performance data and an extensive set of fault management traps to effectively monitor devices and receive notifications on critical system events.

Related Documentation

- SNMP Counters on page 403
- Configuring Media Flow Controller SNMP and SNMP Alarms on page 415
- Configuring the SNMP Agent Using the CLI on page 403

Media Flow Controller MIB Overview

The current Media Flow Controller can be broadly classified as having the following SNMP objects:

- NET-SNMP MIBs that cover the standard SNMP RFCs, including RFC-2863 (IF-MIB), RFC 2790 (HOST-RESOURCES-MIB), RFC-1213 MIB (MIB-II), RFC 4293 (IP-MIB), and RFC 4022 (TCP-MIB).
- A large number of statistical counters for monitoring performance:
  - System-level counters, such as CPU, memory and disk utilization, disk I/O, network utilization, and TCP connections
  - Application-level counters, such as device-level HTTP counters for caching statistics, client- and origin-side connections, bandwidth usage, HTTP transactions, and application errors
  - Namespace-level counters, such as HTTP client and origin server statistics for requests, number of bytes, connections, response code information, and disk usage at the namespace level
- A number of notification objects triggered by critical system-level events that require immediate NOC operator action:
  - Interface or link status changes
  - High usage level of system resources, such as CPU, memory, disk, and network usage
  - Services failures
  - Hardware failures
  - Threshold limit crossing events for caching counters, such as excessive HTTP response errors

Related Documentation

- Media Flow Controller MIB Versions on page 402
- SNMP Support Overview on page 401

Media Flow Controller MIB Versions

Starting with Release 12.2, all the SNMP MIB objects previously defined under two Enterprise OIDs—16858 (Tall Maple) and 32531 (MFD)—have been consolidated under the single Juniper Enterprise OID, which is 2636. Consequently, all these objects were consolidated into the following two MIB files:

- JUNIPER-MFC-MIB (Counters)
- JUNIPER-MFC-NOTIF-MIB (Notifications/Traps)

All the MIB objects defined in these files start with the OID 1.3.6.1.4.1.2636.1.2.

The TMS-MIB and MFD-MIB files are no longer needed and are deprecated.

Related Documentation

- Media Flow Controller MIB Overview on page 401

Configuring the SNMP Agent Using the CLI

You can configure the SNMP agent running in the Media Flow Controller to integrate with third-party network management systems (NMSs). For more details about configuring the SNMP agent using the command-line interface (CLI), see the *Media Flow Controller CLI Command Reference*. The following are the key Media Flow Controller configuration items:

- Configure the NMS IP address.

  snmp host *NMS_IP_address*

- Configure the community string, a shared secret between the NMS and Media Flow Controller.

  snmp host *NMS_IP_address* **traps** *community_string*

- Enable generation of traps in the Media Flow Controller.

  snmp enable traps

- Configure the events for which traps must be generated and sent to the NMS.

  snmp traps event *trap_event*

  See also **snmp traps events** in the *Media Flow Controller CLI Command Reference*.

- In Media Flow Controller Release 12.2.4, 12.3.0, and later, the events listed under the **show snmp events** CLI command are enabled by default and traps are sent when the thresholds for the events are reached. You can disable any events using the **no snmp-server traps event** ***event name*** CLI command.

Related Documentation

- Configuring the SNMP Agent Using the CLI on page 403
- SNMP Support Overview on page 401
- SNMP Counters on page 403

SNMP Counters

The following table lists counters available for use by SNMP systems:

Counter	Description	Type	Category
jmfcSystemVersion	Provides the MFC image version	Snapshot	Device
jmfcSystemId	Provides the MFC Host –id	Incremental	Protocol
jmfcHttpClientRequests	Total number of HTTP client requests received by this SNMP agent	Incremental	Protocol
jmfcHttpClientInBytes	Number of bytes received from HTTP Clients	Incremental	Protocol
jmfcHttpClientOutBytes	Number of bytes servered to the clients	Incremental	Protocol

jmfcHttpClientActiveConns	Number of active HTTP connections	Snapshot	Protocol
mfcHttpClientStatus2xx	Sum of 200 and 206 responses sent to the client	Incremental	Protocol
jmfcHttpClientStatus3xx	Sum of all the 3XX responses sent to the client	Incremental	Protocol
jmfcHttpClientStatus4xx	Sum of all the 4xx reponses sent to the client	Incremental	Protocol
jmfcHttpClientStatus5xx	Sum of all the 5xx reponses sent to the client	Incremental	Protocol
jmfcHttpServerRequests	Number of requests made to origin-servers	Incremental	Protocol
jmfcHttpServerResponses	Number of responses received from Origin servers	Incremental	Protocol
jmfcHttpServerConns	Total number of sessions opened to origin servers	Incremental	Protocol
jmfcHttpServerActiveConns	Number of Active sessions to origin servers	Snapshot	Protocol
jmfcHttpServerInBytes	Total number of bytes received from origin servers	Incremental	Protocol
jmfcHttpServerOutBytes	Total number of bytes sent to origin servers	Incremental	Protocol
jmfcHttpServerStatus3xx	Total number of responses received with a 3xx error code	Incremental	Protocol
jmfcHttpServerStatus4xx	Total number of responses received with a 4xx error code	Incremental	Protocol
jmfcHttpServerStatus5xx	Total number of responses received with a 5xx error code	Incremental	Protocol
jmfcRtspClientRequests	Total number of RTSP client requests received. A single RTSP may result in multiple request exchanges.	Incremental	Protocol
jmfcRtspClientResponses	Total number of RTSP Client responses that was sent to clients. A single RTSP may result in multiple request exchanges. As a result, there may be one or more responses sent to the client for a single session. This field counts each such method response and counts it instead of counting responses for a single object.	Incremental	Protocol
jmfcRtspClientCacheHit	The number of RTSP object requests that resulted in a cache hit, i.e. the object was served from the cache.	Incremental	Protocol
jmfcRtspClientCacheMiss	The number of RTSP object requests that resulted in a cache miss	Incremental	Protocol
jmfcRtspClientCachePartialHit	Number of requests that resulted in a partial hit in cache, and rest of the data was a cache miss	Incremental	Protocol

jmfcRtspClientInBytes	Number of bytes received from RTSP Clients	Incremental	Protocol
jmfcRtspClientOutBytes	Number of bytes sent to RTSP Clients	Incremental	Protocol
jmfcRtspClientConns	Number of RTSP Client sessions	Incremental	Protocol
jmfcRtspClientActiveConns	Number of Active RTSP Client sessions	Snapshot	Protocol
jmfcRtspClientIdleConns	Number of Idle RTSP Client sessions	Snapshot	Protocol
jmfcRtspClientStatus2xx	Number of RTSP requests that were responded with a 2xx status code	Incremental	Protocol
jmfcRtspClientStatus3xx	Number of RTSP requests that were responded with a 3xx status code	Incremental	Protocol
jmfcRtspClientStatus4xx	Number of RTSP requests that were responded with a 4xx status code	Incremental	Protocol
jmfcRtspClientStatus5xx	Number of RTSP requests that were responded with a 5xx status code	Incremental	Protocol
jmfcRtspServerRequests	Number of requests made to origin-servers	Incremental	Protocol
jmfcRtspServerResponses	Number of responses received from Origin servers	Incremental	Protocol
jmfcRtspServerConns	Total number of sessions opened to origin servers	Incremental	Protocol
jmfcRtspServerActiveConns	Number of Active sessions to origin servers	Snapshot	Protocol
jmfcRtspServerIdleConns	Number of Idle sessions to RTSP origin servers	Incremental	Protocol
jmfcRtspServerInBytes	Total number of bytes received from origin servers	Incremental	Protocol
jmfcRtspServerOutBytes	Total number of bytes sent to origin servers	Incremental	Protocol
jmfcRtspServerStatus2xx	The number of RTSP responses received from the origin-server with a 2xx status code	Incremental	Protocol
jmfcRtspServerStatus3xx	The number of RTSP responses received from the origin-server with a 3xx status code	Incremental	Protocol

Disk Table Counters			
jmfcCacheTierUtil	Measures how busy the disk caches are. The value for this counter is a percentage value; the range is 0-100. There are multiple instances of this counter—one for each cache tier (SATA, SAS, and SSD).	1 minute average utilization % (same as LF API counter)	Device
jmfcDiskTable	The tables of disk installed on this system and known by the SNMP agent	Snapshot	Device

jmfcDiskIndex	A unique integer index to identify a disk cache	Not Applicable	Not Applicable
jmfcDiskSerialNumber	The serial number read from the physical disk If no serial number was read from the physical disk, then this field is empty	Snapshot	Device
jmfcDiskName	A symbolic, internally generated cache name for the disk. For example, dc_1, dc_3, etc,	Snapshot/Config value	Device
jmfcDiskType	Tells whether the disk is used as root disk or as caching disk	Configuration value	Device
jmfcDiskTypeModel	The tier type for this disk cache. At present, three tier types are available - SSD(Solid State Device), SAS(Serial Attached SCSI), and SATA(Serial Attached ATA).	Configuration value	Device
jmfcDiskState	A disk cache can be in either of the following states: Cache disabled- In this case, the disk is not used as a cache, and no read or write operations are done to the disk. Cache enabled- in this state, the disk is used as a cache.	Snapshot	Device
jmfcDiskBlockSize	The block size configured for this disk cache. For example, a value of 2048 denotes 2 megabytes.	Snapshot	Device
jmfcDiskTotalSize	The total cache -able size of the disk. This does not include the space reserved for meta -data.	Snapshot	Device
jmfcDiskBlocksUsed	The number of blocks used for storing objects that are cached. Objects that have expired are NOT accounted for.	Snapshot	Device
jmfcDiskFreeSpace	Total free space that is available for caching objects. Objects that have expired are also counted as space that is free and available for caching.	Snapshot	Device
jmfcDiskCapacity	Total storage space available on the disk	Snapshot	Device
jmfcDiskUtil	Disk utilization in percentage	Snapshot	Device
jmfcDiskIORate	Average read/write per second taken from iostat	1 second average	Device
jmfcDiskIORateErrThreshold	User configured error threshold value on the IO rate .Maximum value beyond which notification is generatated.	Config value	Device
jmfcDiskIORateClrThreshold	User configured threshold value for clearing the notification when IO rate falls below this value	Config value	Device
jmfcDiskSpaceErrThreshold	User configured error threshold for disk space availability	Config value	Device

jmfcDiskSpaceClrThreshold	User configured clearing threshold for disk space availability	Config value	Device
jmfcDiskBW	The bandwidth usage of the disk	Snapshot	Device
jmfcDiskBWErrThreshold	The error threshold for the disk bandwidth	Config value	Device
jmfcDiskBWClrThreshold	The clearing threshold for the disk bandwidth	Config value	Device
jmfcRootDiskMirrorComplete	The notification sent when the root disk re-mirroring is complete and contains information about the root disk that was re-mirrored.	Not applicable	Device
jmfcRootDiskMirrorBroken	The notification sent when the root disk mirror is broken and contains information about what caused the root disk mirror to break.	Not applicable	Device
jmfcInterfaceTable	List of ether channel interfaces	Not Applicable	Not Applicable
jmfcIfIndex	A unique value, greater than zero, for each interface	Not Applicable	Not Applicable
jmfcIfName	Interface name	Config value	Interface
jmfcIfSpeed	The speed supported by the interface	Snapshot	Interface
jmfcIfAdminStat	Indicates whether the interface was turned on by the admin	Config value	Interface
jmfcIfOperStat	Indicates whether the interface link is turned on and active	Snapshot	Interface
jmfcNetIntfUtilCurrent	Provides the overall network utilization in percentage	Snapshot	Interface
jmfcNetIntfUtilErrThreshold	Error threshold for triggering notification when the network utilization goes beyond this configured value	Config value	Interface
jmfcNetIntfUtilClrThreshold	Clearing threshold for notification when then network utilization goes below this configured value	Config value	Interface

Namespace Table Counters			
jmfcNamespaceTable	Provides the list of namespaces	Not Applicable	Not Applicable
jmfcNamespaceIndex	Unique id for each namespace	Not Applicable	Not Applicable
jmfcNamespaceName	Name of the namespace	Config value	Namespace
jmfcNamespaceUID	Unique UID of the namespace	Config value	Namespace

jmfcNamespaceStatus	Current status of the namespace – active/inactive	Snapshot	Namespace
jmfcNamespaceHttpClientTable	Provides the Namespace client details	Not Applicable	Not Applicable
jmfcNamespaceHttpClientRequests	Total number of HTTP client requests received	Incremental	Namespace
jmfcNamespaceHttpClientResponses	Total number of HTTP Client responses that was sent to clients	Incremental	Namespace
jmfcNamespaceHttpClientCacheHit	The number of client HTTP requests that resulted in a cache hit, i.e. the entire object was served from the cache.	Incremental	Namespace
jmfcNamespaceHttpClientCacheMiss	The number of HTTP client requests that resulted in a complete cache-miss, i.e. the object was not in cache and was served from a origin server	Incremental	Namespace
jmfcNamespaceHttpClientInBytes	Number of bytes received from HTTP Clients	Incremental	Namespace
jmfcNamespaceHttpClientOutBytes	Number of bytes sent to HTTP Clients	Incremental	Namespace
jmfcNamespaceHttpClientConns	Number of HTTP Client sessions that are open	Incremental	Namespace
jmfcNamespaceHttpClientActiveConns	Number of HTTP Client sessions that are active	Snapshot	Namespace
jmfcNamespaceHttpClientIdleConns	Number of HTTP client connections that are open but idle	Snapshot	Namespace
jmfcNamespaceHttpClientStatus2xx	The number of HTTP client requests that were responded with a 2xx status code	Incremental	Namespace
jmfcNamespaceHttpClientStatus3xx	The number of HTTP client requests that were responded with a 3xx status code	Incremental	Namespace
jmfcNamespaceHttpClientStatus4xx	The number of HTTP client requests that were responded with a 4xx status code	Incremental	Namespace
jmfcNamespaceHttpClientStatus5xx	The number of HTTP client requests that were responded with a 5xx status code	Incremental	Namespace
jmfcNamespaceHttpClient404Count	Number of requests resulting in HTTP 404 Not Found	Incremental	Namespace
jmfcNamespaceHttpClient302Count	Number of requests resulting in HTTP 302 Redirect	Incremental	Namespace
jmfcNamespaceHttpClient304Count	Number of requests resulting in 304 Not Modified	Incremental	Namespace

jmfcNamespaceHttpClient206Count	Number of requests resulting in HTTP 206 Partial Content	Incremental	Namespace
jmfcNamespaceHttpClient200Count	Number of requests resulting in HTTP 200 OK	Incremental	Namespace
jmfcNamespaceHttpServerTable	This table provides HTTP specific origin-server counters for a given namespace. The namespace is identified by the index identified in the jmfcNamespaceTable	Not Applicable	Not Applicable
jmfcNamespaceHttpServerRequests	Total number of HTTP server requests made to origin servers	Incremental	Namespace
jmfcNamespaceHttpServerResponses	Total number of HTTP server responses that were received from origin servers	Incremental	Namespace
jmfcNamespaceHttpServerInBytes	Number of bytes received from HTTP origin servers	Incremental	Namespace
JmfcNamespaceHttpServerOutBytes	Number of bytes sent to HTTP origin servers	Incremental	Namespace
jmfcNamespaceHttpServerConns	Number of HTTP origin server sessions	Incremental	Namespace
jmfcNamespaceHttpServerActiveConns	Number of Active HTTP server sessions	Snapshot	Namespace
jmfcNamespaceHttpServerIdleConns	Number of Idle HTTP server sessions	Snapshot	Namespace
jmfcNamespaceHttpServerStatus2xx	Total number of HTTP responses received from origin servers with a 2xx status code	Incremental	Namespace
jmfcNamespaceHttpServerStatus3xx	Total number of HTTP responses received from origin servers with a 3xx status code	Incremental	Namespace
jmfcNamespaceHttpServerStatus4xx	Total number of HTTP responses received from origin servers with a 4xx status code	Incremental	Namespace
jmfcNamespaceHttpServerStatus5xx	Total number of HTTP responses received from origin servers with a 5xx status code	Incremental	Namespace
jmfcNamespaceRtspClientTable	This table provides details on RTSP client statistics for a namespace	Incremental	Namespace
jmfcNamespaceRtspClientRequests	Total number of RTSP client requests received for this namespace	Incremental	Namespace
jmfcNamespaceRtspClientResponses	Total number of RTSP Client responses that was sent	Incremental	Namespace
jmfcNamespaceRtspClientCacheHit	Total number of requests that resulted in a cache hit, i.e. the object was served from the cache.	Incremental	Namespace

jmfcNamespaceRtspClientCacheMiss	Total number of client requests that resulted in a cache miss, i.e. the object was not found in cache and was served from the origin-server.	Incremental	Namespace
jmfcNamespaceRtspClientCachePartialHit	Number of requests that resulted in a partial hit in cache, and rest of the data was a cache miss	Incremental	Namespace
jmfcNamespaceRtspClientInBytes	Number of bytes received from RTSP Clients	Incremental	Namespace
jmfcNamespaceRtspClientOutBytes	Number of bytes sent to RTSP Clients	Incremental	Namespace
jmfcNamespaceRtspClientConns	Number of RTSP Client sessions	Incremental	Namespace
jmfcNamespaceRtspClientActiveConns	Number of Active RTSP Client sessions	Snapshot	Namespace
jmfcNamespaceRtspClientIdleConns	Number of Idle RTSP Client sessions	Snapshot	Namespace
jmfcNamespaceRtspClientStatus2xx	Number of RTSP requests that were responded with a 2xx status code	Incremental	Namespace
jmfcNamespaceRtspClientStatus3xx	Number of RTSP requests that were responded with a 3xx status code	Incremental	Namespace
jmfcNamespaceRtspClientStatus4xx	Number of RTSP requests that were responded with a 4xx status code	Incremental	Namespace
jmfcNamespaceRtspClientStatus5xx	Number of RTSP requests that were responded with a 5xx status code	Incremental	Namespace
jmfcNamespaceRtspServerTable	This table provides RTSP server specific statistics for a namespace.		
jmfcNamespaceRtspServerRequests	Total number of RTSP requests that were sent to an RTSP origin server	Incremental	Namespace
jmfcNamespaceRtspServerResponses	Total number of RTSP Server responses that was received from an origin server	Incremental	Namespace
jmfcNamespaceRtspServerInBytes	Number of bytes received from RTSP origin servers	Incremental	Namespace
jmfcNamespaceRtspServerOutBytes	Number of bytes sent to RTSP origin servers	Incremental	Namespace
jmfcNamespaceRtspServerConns	Number of RTSP origin server sessions	Incremental	Namespace
jmfcNamespaceRtspServerActiveConns	Number of Active RTSP server sessions	Snapshot	Namespace
jmfcNamespaceRtspServerIdleConns	Number of Idle RTSP server sessions	Snapshot	Namespace

jmfcNamespaceRtspServerStatus2xx	Total number of responses received from RTSP origin servers with a 2xx status code	Incremental	Namespace
jmfcNamespaceRtspServerStatus3xx	Total number of responses received from RTSP origin servers with a 3xx status code	Incremental	Namespace
jmfcNamespaceRtspServerStatus4xx	Total number of responses received from RTSP origin servers with a 4xx status code	Incremental	Namespace
jmfcNamespaceRtspServerStatus5xx	Total number of responses received from RTSP origin servers with a 5xx status code	Incremental	Namespace
jmfcNamespaceResourceTable	This table provides the resource usage statistics for a namespace.	Not Applicable	Not Applicable
jmfcNamespaceDiskCacheUsed	Amount of megabytes used in storing objects in disk-caches by this namespace	Snapshot	Namespace
jmfcNamespaceObjects	Number of objects cached for this namespace	Snapshot	Namespace
jmfcNamespaceDiskCacheUsedTier1	Total bytes used in disk caches at tier-1	Snapshot	Namespace
jmfcNamespaceDiskCacheUsedTier2	Total bytes used in disk caches at tier-2	Snapshot	Namespace
jmfcNamespaceDiskCacheUsedTier3	Total bytes used in disk caches at tier-3	Snapshot	Namespace
jmfcNamespaceTunnelledBytes	Total number of tunneled bytes (ClientIn and ClientOut)	Incremental	Namespace
jmfcNamespaceTunnelledRequests	Total number of tunneled requests	Incremental	Namespace

Service Table Counters			
jmfcServiceTable	Provides the list of services managed by the Process manager	Not Applicable	Not Applicable
jmfcServiceIndex	A unique id for the Service entry in the table	Not Applicable	Not Applicable
jmfcServiceName	Serivce name	Config	System
jmfcServiceStatus	The current status of the service- running, stopped etc	Snapshot	System
jmfcServiceNumFailures	The number of times process has crashed or exited unexpectedly	Incremental	System
jmfcServicePid	The current PID of the service	Snapshot	System
jmfcServiceLastDown	The last time stamp at which the service was down	Snapshot	System

jmfcServiceLastUp	The timestamp at which the service was last started	Snapshot	System
jmfcServiceCrashReason	A description of why the service crashed. Sent along with the notification for the service crash.	Description/Snapshot	System
jmfcServiceDowntime	The time in seconds for which the service is down	Snapshot	System

CPU Table Counters			
jmfcCpuTable	Provides the list of CPU in the system	Not Applicable	Not Applicable
jmfcCpuIndex	Provides the unique id for each CPU entry	Not Applicable	Not Applicable
jmfcIdleTime	Time in milliseconds CPU has spent idle	Snapshot	System
jmfcSystemTime	Time in milliseconds CPU has spent busy with system tasks	Snapshot	System
jmfcUserTime	Time in milliseconds CPU has spent busy with user tasks	Snapshot	System
jmfcUtilPct1Min	Percentage CPU utilization over the past minute	Snapshot	System
jmfcCpuUtil	List of all cpu Util Values for all cores. "glob_lf_cpu_util" will be used directly.	1 minute average	Device Level
jmfcCpuUtilErrThreshold	Limit on cpu utilization beyond which error is reported	Config	Device Level
jmfcCpuUtilClrThreshold	Clearing threshold for cpu utilization	Config	Device Level
jmfcAppMaxCpuUtil	Max CPU util value of the cores running the HTTP application. The global variable "glob_lf_app_cpu_util" will be used directly.	1 minute average	Protocol
jmfcAppMaxCpuUtilErrThreshold	Configured upper threshold on HTTP application CPU utilization for triggering an alarm	Config	Protocol
jmfcAppMaxCpuUtilClrThreshold	Configured threshold value on HTTP application CPU utilization to clear the alarm	Config	Protocol
jmfcMaxThroughput	Rated possible maximum throughput of the Media Flow Controller	Snapshot (same as LF API counter)	Device level
jmfcMaxOpenConnections	Rated possible maximum of open connections for the Media Flow Controller	Snapshot (same as LF API counter)	Device level
jmfcMemUtil	Non cache RAM memory utilization percentage	1 minute average	Device Level

jmfcMemUtilErrThreshold	Configured value for memory utilization beyond which error is reported	Config	Device Level
jmfcMemUtilClrThreshold	Clearing threshold value for memory utilization	Config	Device level
jmfcPagingCurrent	Current paging rate of the system	Snapshot	Device level
jmfcPagingCurrent	Current paging rate of the system	Snapshot	Device level
jmfcPagingErrThreshold	Threshold value for paging beyond which the paging value generates error notification	Config	Device level
jmfcPagingClrThreshold	Clearing threshold value for paging	Config	Device level
jmfcOriginBw	Provides the bandwidth usage with origin. The counter “glob_tot_size_from_origin” value is sampled and chd is computed.	Average	Resource
jmfcOriginBwErrThreshold	User configured error limit for origin bandwidth usage	Config	Resource
jmfcOriginBwClrThreshold	Use configured clearing threshold for origin bandwidth usage	Config	Resource
jmfcCacheBwErrThreshold	User configured error limit for cache bandwidth usage	Config	Resource
jmfcCacheBwClrThreshold	User configured clearing threshold for cache bandwidth usage	Config	Resource
jmfcConnectionRateCurrent	Current value of TCP network connections per second. The variable : "glob_tot_http_sockets" is sampled at interval of 1 sec. Then the chd value is computed using the samples.	1 second average	Resource
jmfcConnectionRateErrThreshold	Error threshold for connection rate	Config	Resource
jmfcConnectionRateClrThreshold	Clearing threshold for the connection rate	Config	Resource

Resource pool table counters			
jmfcResourcePoolTable	List of resource pool configured	Not Applicable	Not Applicable
jmfcResourcePoolIndex	Unique id for each resource pool added	Not Applicable	Not Applicable
jmfcResourcePoolName	Resource pool name	Config Name	Resource Pool
jmfcResourcePoolBandwidth	Delivery throughput represented in Mbps	Snapshot (Same as LF API counter. It is the most recent value sent by LF API)	Resource Pool

jmfcResourcePoolProvisionedBandwidth	Configured max. bandwidth	Snapshot	Resource Pool
jmfcResourcePoolActiveConns	Number of active TCP sessions established by users	Snapshot (same as LF API counter. It is the most recent value sent by LF API)	Resource Pool
jmfcResourcePoolProvisionedMaxConnections	Configured Max Connections value	Snapshot	Resource Pool
jmfcResourcePoolNamespaceList	MIB variable giving the mapping of a resource pool and the list of namespaces bound to that resource pool. Can be added to the mfdResourcePoolTable	Snapshot	Resource Pool
jmfcResourcePoolNamespaceIndexList	MIB variable giving the mapping of a resource pool to the namespace indexes bound to that resource pool. The index should be the same used in the jmfcNamespaceTable. Can be added to the mfdResourcePoolTable	Snapshot	Resource Pool

Cluster table counters			
jmfcClusterTable	List of clusters configured	Not Applicable	Not Applicable
jmfcClusterIndex	Unique ID for each cluster entry	Not Applicable	Not Applicable
jmfcClusterName	Cluster name	Config	Cluster level
jmfcClusterType	Type of the cluster configured	Config	Cluster level
Node table			
jmfcNodeTable	List of all the nodes	Not Applicable	Not Applicable
jmfcNodeIndex	Unique id for each node name	Not Applicable	Not Applicable
jmfcNodeName	Name of the node that is part of the cluster	Config	Device
jmfcNodeStatus	Status of the node – whether it is up or down	Snapshot	Device
jmfcTransactionRateCurrent	Current value of HTTP transactions done per second. The global variable "glob_http_tot_transactions" is used for sampling and chd calculation. The chd value is compared with the threshold	1 second average	Protocol
jmfcTransactionRateErrThreshold	Max value beyond which error is reported for HTTP transaction rate	Config	Protocol
jmfcTransactionRateClrThreshold	Clearing threshold for HTTP transaction rate	Config	Protocol

jmfcCacheHitRatio	Cache hit ratio of the system	Average	Resource
jmfcCacheHitRatioErrThreshold	Error threshold for the cache hit ratio	Config	Resource
jmfcCacheHitRatioClrThreshold	Clearing threshold for the cache hit ratio	Config	Resource

Logger table			
jmfcLoggerTable	Table maintaining list of access log profile configuration. The field LoggerName and Logger Server are used when sending the logExport failed notification	Not Applicable	Not Applicable
jmfcLoggerIndex		Not Applicable	Not Applicable
jmfcLoggerName		Config	Device
jmfcLoggerTemplate		Config	Device
jmfcLoggerServer		Config	Device

Related Documentation

- SNMP Support Overview on page 401

Configuring Media Flow Controller SNMP and SNMP Alarms

Media Flow Controller has SNMP agents that can respond to SNMP operations. An SNMP manager can ask Media Flow Controller for information and generate notifications of SNMP events to configured notify recipients.

- Configuring SNMP on page 415
- snmp traps events on page 417
- SNMP Alarms on page 417

Configuring SNMP

Configuring SNMP on Media Flow Controller consists of enabling the SNMP server, configuring community strings, traps, traps events, and hosts (also referred to as trap sinks) to receive traps, as well as authentication parameters.

To configure SNMP:

1. Configure SNMP hosts, or trap sinks, to receive SNMP query responses, or traps, sent from Media Flow Controller. When configured, the host is enabled, and you can disable it with **snmp-server host *IP_address* disable** to temporarily stop Media Flow Controller from sending traps to it. You need to know the trap **version** and **community** for the traps to be sent to that host; the **community** identifies who can query.

   snmp-server host *IP_address*
   snmp-server host traps version *version community_string*

The traps associated with the specified version and community are sent to the specified host.

2. Enable community-based authentication if you intend to use communities other than the default **public** community, and the sending of traps (traps are not sent until a host is configured to receive them). By default, the Media Flow Controller SNMP server function is enabled; you can disable it with **no snmp-server enable**, which stops the serving of SNMP variables and sending of traps.

   snmp-server enable communities
   snmp-server enable traps

   The default community, **public**, is enabled and you can configure additional communities. The following traps are sent to the enabled host configured for the traps and communities: **cold boot** (can include SNMP configuration changes), **link up** or **down**, **CPU load too high**, **paging activity too high**, **a process crash**, and **a process unexpected exited**.

3. Create SNMP communities that determine which agents can make queries to which SNMP servers. There are some predefined communities, such as the default **public** community, or you can create a new community. If the community name contains spaces, enclose it in quotes. Media Flow Controller only allows read-only, **ro**, communities that permit only SNMP queries, or GETs. SNMP SETs for configuration changes are not allowed. Only the agents querying from a community configured in the Media Flow Controller SNMP server community list can make queries to that Media Flow Controller SNMP server. To enable read-only, **ro**, communities:

   snmp-server community *community_string* ro

   The traps associated with that community are sent to the **host**, or trap sink, that is enabled and configured with that community and for those traps.

4. Configure SNMP listen interfaces to restrict access to a configured list of interfaces. By default, list is enabled. When you add an interface, SNMP queries are restricted to the interface or interfaces in the listen interfaces list. These interfaces should be statically configured with DHCP and zeroconf disabled. You can disable the list with **no snmp-server listen enable**. To add listen interfaces:

   snmp-server listen *interface_ID*

   Only the configured interfaces listen for SNMP queries. If the configured interface is also running as a DHCP client, it does not listen for SNMP queries. If DHCP is disabled on this interface, the configured interface begins to listen to SNMP queries.

5. Set the SNMP server contact and location. These are the syscontact and syslocation variables served from the system MIB in MIB-II.

   snmp-server contact *contact_name*
   snmp-server location *location_of_system*

   The manager of this SNMP server and the SNMP server location are identified.

6. Specify the types of events to send as SNMP traps. By default, the entire list of notifiable events is sent as SNMP traps to any configured trap sinks. For more details about configuring the SNMP notification traps using the command-line interface (CLI), see the *Media Flow Controller CLI Command Reference*.

snmp-server traps event *event_name*

Only the configured trap events are sent.

7. Specify the trap recipients. See "Configuring Media Flow Controller Fault Notifications (CLI)" on page 372 for details on additional options.

   email notify recipient *email_address*

   Traps events are sent to the configured e-mail address.

snmp traps events

Media Flow Controller generates a number of traps to notify you about critical system events. You can configure Media Flow Controller to send SNMP traps to a third-party network management system. For details about configuring the SNMP notification traps using the command-line interface (CLI), see the *Media Flow Controller CLI Command Reference*.

SNMP Alarms

This section describes the SNMP alarms that can reach you through a configurable **email event name** option.

Media Flow Controller supports the **Entity** and **Asset** SNMP MIBs for discovery, asset management, alarms, and traps; that is HOST-RESOURCES, HOST-RESOURCES-TYPES, IF, IP, IDP, and TCP MIBs. To view these MIBs, go to the Customer Support website.

Typically, SNMP uses User Datagram Protocol (UDP) ports 161 for the agent and port 162 for the manager. The Manager can send requests from any available port (source port) to port 161 in the agent (destination port). The agent response is returned to the source port. The Manager receives traps on port 162. The agent can generate traps from any available port.

NOTE: **Media Flow Controller does not support provisioning over SNMP V3.**

Related Documentation

- Configuring the SNMP Agent Using the CLI on page 403
- SNMP Support Overview on page 401
- SNMP Counters on page 403

CHAPTER 14

Troubleshooting Media Flow Controller

Viewing Information Using Show Commands

Problem Find information about Media Flow Controller performance issues.

Use the Media Flow Controller CLI **show** commands to find the system information you need. This section excludes commands that show settings that are not useful for fault management, including **show** commands for **banner**, **cli**, **clock**, **delivery**, **email**, **ip**, **license**, **logging**, **ntp**, **radius-server**, **snmp-server**, **ssh**, **tacacs-server**, **telnet**, **terminal**, **username**, **web**, and **whoami**. See Table 40 on page 374 and Table 41 on page 377 for descriptions of HTTP response codes and sub-codes.

Solution **show analytics**—Display the analytics manager information (default settings shown):

```
Analytics Configuration:
   Cache Promotion                      : Disabled
   Cache Promotion Size threshold       : 0 (All objects are promoted)
   Cache ingest policy                  : Hybrid
   Memory Limit                         : AUTO( 0 MiB)
```

You can configure some of these settings with **analytics** commands.

show configuration—Lists the CLI commands needed to bring the state of a fresh system up to match the current persistent (saved) state of this system. A short header is included, containing the name and version number of the configuration. Because commands that have not been saved, or would return a setting to its default, are not included, this command on a fresh configuration produces no output, except the header. It includes the following arguments:

- **files [*filename*]**—If no *filename* is specified, lists the configuration files in persistent storage. If *filename* is specified, lists the commands to re-create the configuration in that file; only non-default commands are shown.
- **full**—Same as **show configuration** but includes commands that set default values.
- **running**—Same as **show configuration** except that it applies to the currently running configuration, rather than the active saved configuration.
- **text files**—Lists text-based configuration files.

show counters—Lists the following information. See "Media Flow Controller Log Codes and Sub-Codes" on page 374, for descriptions of HTTP response codes.

Field	Description
Total Number of Open Connections	Lists total number of all open connections.
Total Number of Open Server Connections	Lists total number of open server connections.
Total Bytes Served from RAM cache	Lists the total number of bytes retrieved from MFC RAM memory.
Origin Server Counters	
Total Bytes served from Origin Server	Lists the total number of HTTP, NFS, and RTSP bytes received from the origin (remote) server, including headers.
Total Bytes served from HTTP Origin Server	Lists the total number of HTTP bytes received from the remote server, including headers.
Total Bytes served from NFS Origin Server	Lists the total number of NFS bytes received from the remote server, including headers.
Total Bytes served from RTSP Origin Server	Lists the total number of RTSP bytes received from the remote server, including headers.
Total Bytes served from Disk cache	Lists the number of bytes retrieved from physical memory.

Field	Description
Total Bytes served	Lists the total number of bytes served by the MFC server.
Ingest Fetch Count	Lists the number of origin transactions completed, including ingest and client delivery transactions..
Total Disk Read Operations	Lists the total number of disk read operations in the transaction.
Total Disk Write Operations	Lists the total number of disk write operations in the transaction.
Virtual Player Number of Seeks	Lists the total number of seek actions in the transaction.
Virtual Player Hash Verification Failed Errors	Lists the total number of hash verification errors encountered.
Total Number of ports	Lists the number of ports used.
Active Connections on Port eth0	Lists the current number of connections on the indicated port.
Active Connections on Port lo	Lists the current number of connections on the indicated port.
Number of Requests on Namespace <namespace>	Lists the total completed HTTP transactions under the indicated namespace.
Total Number of HTTP 0.9 requests	Lists the total number of 0.9 requests.
Total number of Active HTTP Connections	Lists the total number of active HTTP connections.
Total number of HTTP Connections	Lists the total number of HTTP connections.
Total number of HTTP Transactions	Lists the total number of completed HTTP transactions.
Total number of HTTP 200 responses	Lists the total number of the indicated type of HTTP responses.
Total number of HTTP 206 responses	Lists the total number of the indicated type of HTTP responses.
Total number of HTTP 302 responses	Lists the total number of the indicated type of HTTP responses.
Total number of HTTP 304 responses	Lists the total number of the indicated type of HTTP responses.
Total number of HTTP 400 responses	Lists the total number of the indicated type of HTTP responses.
Total number of HTTP 404 responses	Lists the total number of the indicated type of HTTP responses.
Total number of HTTP 416 responses	Lists the total number of the indicated type of HTTP responses.
Total number of HTTP 500 responses	Lists the total number of the indicated type of HTTP responses.
Total number of HTTP 501 responses	Lists the total number of the indicated type of HTTP responses.

Field	Description
Total number of HTTP 503 responses	Lists the total number of the indicated type of HTTP responses.
Total number of HTTP 504 responses	Lists the total number of the indicated type of HTTP responses.
Total number of HTTP Timeouts	Lists the total number of HTTP timeouts.
Total HTTP Well finished count	Lists the number of transactions completed.
Total Number of Cache-hits	Lists the total number of hits.
Total Number of Cache-miss	Lists the total number of misses.
HTTP TUNNEL STATS	
Total Connections	Lists the total number of connections.
Total Active Connections	Lists the total number of active connections.
Total Bytes Served	Lists the total number of bytes served.
Total Errors	Lists the total number of errors encountered.
ERROR COUNTERS	
Number of Scheduler Errors on get data	Lists the number of errors encountered during the get data operation.
Number of HTTP deadline missed tasks	Lists the number of tasks missing the deadline.
PROXY ERRORS	
HTTP Parse Error Count	Lists the number of parse errors encountered.
Cluster L7 statistics	
Total number of Redirects	Lists the number of redirects.
Total number of Redirect errors	Lists number of redirect errors.
Total number of Active RTSP Connections	Lists the total active Real Time Streaming Protocol connections.
Total number of RTSP Transactions	Lists total number of Real Time Streaming Protocol transactions.
Total Number of RTSP Packets Forwarded	Lists total number of Real Time Streaming Protocol packets forwarded.

Field	Description
Total number of RTSP 200 responses	Lists total number of Real Time Streaming Protocol type 200 responses.
Total number of RTSP 400 responses	Lists total number of Real Time Streaming Protocol type 400 responses.
Total number of RTSP 404 responses	Lists total number of Real Time Streaming Protocol type 404 responses.
Total number of RTSP 500 responses	Lists total number of Real Time Streaming Protocol type 500 responses.
Total number of RTSP 501 responses	Lists total number of Real Time Streaming Protocol type 501 responses.
RTCP Statistics	
Total Number of RTCP Packets Forwarded	Lists the number of RTCP Packets forwarded.
Total Number of Sender Report sent	Lists the number of Sender Reports sent.
Total Number of Receiver Report sent	Lists the number of Receiver Reports sent.
Total Number of RTCP Packets Received	Lists the number of RTCP packets received.
Total Number of Sender Report Received	Lists the number of Sender Reports received.
Total Number of Receiver Report Received	Lists the number of Receiver Reports received.
PROXY ERRORS	
OM Error Connection Failed Count	Lists the number of origin Connection Failed errors.
RTSP Parse Error Count	Lists the number of Real Time Streaming Protocol errors encountered.
Disk Cache Counters	
Total Objects	Lists the cache objects total.
Total Objects cached on SAS	Lists the number of objects in the SAS cache.
Total Objects cached on SSD	Lists the number of objects in the SSD cache.
Total Objects cached on SATA	Lists the number of objects in the SATA cache.

show files—Lists available files or their counters. Includes the following arguments:

- **debug-dump**—Lists debug dump files.
- **stats**—Lists statistics reports.
- **system**—File system information; includes the following statistics for **/config** and **/var**:
  - Bytes Total
  - Bytes Free
  - Bytes Used
  - Bytes Available
  - Bytes Percent Free
  - Inodes Total
  - Inodes Free
  - Inodes Used
  - Inodes Percent Free
- **tcpdump**—Lists **tcpdump** files.

show hosts—Displays the hostname, DNS configuration, and static host mappings.

show interface [configured]—Displays Information about each system interface, including:

- **Admin** state—Whether or not the interface is enabled.
- **Link** state—**Up** means a cable is plugged into that interface and it is "live," or connected to equipment that is turned on at the other side; for example, a switch, router, or another computer.
- Current **TX** (transmissions out) and **RX** (transmissions received) statistics.

  Use **configured** to see the settings of the interfaces, rather than their runtime state.

show log—View event logs, including commands executed during this session.

show media-cache disk—The **list** argument lists available caches and information on each, including **Device** (name), **Type**, (cache), **Tier** (1=SSD, 2=SAS, 3=SATA), **Active** status, **Enabled** status, **Free Space**, and **Disk State**. Use **show media-cache disk** ***disk_name*** for details on a particular disk.

show memory—Shows memory usage, including total, used, and free for physical and swap.

show namespace (list | ***namespace_name*** **[counters | object])**—Gets namespace-specific information:

- **list**—Lists the configured namespaces and their UIDs.
- ***namespace_name***—Lists settings for the specified namespace.
- ***namespace_name*** **object list (all | uri | pattern)**—Lists objects in the specified namespace.
- ***namespace_name*** **counters**—Lists the specified namespace's counters, including:

```
HTTP Resource Monitoring Counters:
Client Active Sessions:
Current Bandwidth:
Served Bytes:
Transaction Per sec:
Cache Hit Ratio(Bytes):
Cache Hit Ratio(Req):
HTTP Client Counters:
Number of requests:
Number of responses:
Total Bytes Received:
Total Bytes Sent:
From Memory Cache:
From Disk Cache:
From Origin:
Cache Hits:
Cache Misses:
Responses with 2xx status code:
Responses with 3xx status code:
Responses with 4xx status code:
Responses with 5xx status code:
HTTP Origin Counters:
Number of requests:
Number of responses:
Total Bytes Received:
Total Bytes Sent:
Responses with 2xx status code:
Responses with 3xx status code:
Responses with 4xx status code:
Responses with 5xx status code:
```

show network—Displays network configurations, including time out, maximum connections, , and session maximum bandwidth settings.

show ram-cache —Displays the buffer-manager configuration; defaults are sync-interval is **86400 seconds**, object minimum size is **0** (zero), revalidate-window is **120 seconds**, unsuccessful response RAM cache percentage is **0** (zero), and maximum RAM cache size is **0** (zero) or **AUTO**. Also lists the current RAM cache size.

show resource-pool global-pool —Lists the currently available system-wide resources, including maximum allowed concurrent sessions, maximum available bandwidth, and cache tier memory limits per disk.

show service—Displays the configuration and status (current status, number of failures, last terminated, and uptime) for these services: Media Flow Publisher file publishing (**mod-file-publisher**), Media Flow Publisher live publishing (**mod-live-publisher**), access log and error log (**mod-log**), offline origin manager (**mod-oom**), delivery (**mod- delivery**).

show statistics—Displays key statistics; including:

- **Current Bandwidth (MB/Sec)**—The rate at which Media Flow Controller is currently delivering the service.
- **Current Cache Bandwidth (MB/Sec**)—The delivery bandwidth at which Media Flow Controller is delivering from cache, excluding direct deliveries from the origin.

- **Current Disk Bandwidth (MB/Sec)**—The delivery bandwidth coming from objects in the disk; should be a subset of Current Cache Bandwidth.
- **Current Origin Bandwidth (MB/Sec)**—The delivery bandwidth for objects being fetched from the origin and directly delivered.
- **Avg Number of Connections Per Sec**—On average, the connection accept rate.
- **Avg HTTP Transactions per Sec**—On average, the number of completed HTTP transactions.
- **Avg Cache Bandwidth (MB/Sec)**—On average, the data fetch rate from buffer to network for delivery.
- **Avg Disk Bandwidth (MB/Sec)**—On average, the data fetch rate from disk.
- **Avg Origin Bandwidth (MB/Sec)**—On average, the data fetch rate from origin (happens only when there is a cache miss).
- **Avg Proxy Rate (Origin Manager) (MB/Sec)**—Total bytes received from the origin, sampled every 5 seconds and averaged since Media Flow Controller start.
- **Current Proxy Rate in this sec (MB/Sec)**—Rate of GET requests made in one second.
- **Per Disk Bandwidth (MB/Sec)**—Lists available disks and the bandwidth for each.
- **Per port statistics (TX Bytes Per Sec on *interface*)**—Lists transferred bytes per second for each interface.

show stats *type*—Lists statistics (stats) settings and gathered data; includes the following statistic types (see **stats** in the *Media Flow Controller CLI Command Reference* for CLI details on alarm, chd, and sample types):

- **alarm**—Lists status and configuration of statistics-based alarms.
- **chd**—Lists configuration of statistics CHDs.
- **cpu**—Lists CPU statistics.
- **sample**—Lists configuration of statistics samples.

show system—Lists system configuration (debug level and mod).

show users—Lists information about user logins.

show version—Lists version information for the current system image.

show virtual-player [list] [*name*]—Lists the virtual player settings; use **list** to see a list of defined virtual players and their type; use ***name*** to see the settings.

Related Documentation

- Media Flow Controller Log Codes and Sub-Codes on page 374
- About the Media Flow Controller CLI on page 21

Internal Watchdog

Media Flow Controller has a watchdog that monitors the service layer, including monitoring the health of data delivery to users, origin fetch, and disk functions. This internal watchdog mechanism regularly checks the proper functioning of the service layer and restarts the service if the check fails. This ensures that Media Flow Controller only loses service for a short duration if the service layer encounters a defect.

As a part of this feature, there is a default namespace called **mfc_probe**. This namespace is configured as a reverse proxy with the internal management Web server as its origin server. There is a canned object file in this origin server that you can request to trigger a full test of the service path. You can also use this to configure any probes to request this canned object; for example, load balancers typically expect probes to be configured and the **mfc_probe** namespace can be used for this.

NOTE: **Do not make changes to the watchdog/mfc_probe namespace unless instructed by technical support .**

The probe URI is:

http://*MFC_IP*/mfc_probe/mfc_probe_object.dat

The Media Flow Controller watchdog probes the service layer by sending an HTTP request (probe) to the default **mfc_probe** namespace every 5 seconds. If a response arrives within the timeout period, the service layer is alive. If there is no response three times in a row, Media Flow Controller determines that the service layer is stuck and restarts it. While restarting the service layer, a core file is generated at /coredump/snapshots.

Related Documentation

- Understanding Authentication, Authorization, and User Options on page 46
- Viewing Information Using Show Commands on page 419

Testing Network Connectivity

Problem To make sure your network connections are configured and behaving properly.

Solution

- Confirm that your computer has the appropriate settings for DNS servers, the correct hostname, an available IP address with the proper subnet mask (**show hosts**), and the proper default gateway (**show ip default-gateway**).
- Check network interfaces (**show interfaces**). Make sure the links are up, the addresses correct, and the default routes are set correctly (**show ip route**). Also check cable connectivity.
- Check ARP tables (**show arp**).
- Check DNS or hostname configurations (**show hosts**).
- Check that applications such as SSH is working.

- Test connections to remote servers by using **ping** (Ctrl+c to stop ping):
  a. Ping the loopback address (by using the ping 127.0.0.1 command) to verify that TCP/IP is installed and working correctly on the local computer.
  b. Ping the local computer IP address to verify it was added to the network correctly.
  c. Ping the IP address of the default gateway to verify that the gateway is functional and it is possible to connect to a local host on the local network.
  d. Ping the IP address of another remote host to verify that you can communicate through a router.
- Check the network path to a destination with **traceroute**:
  a. At the command prompt, enter **traceroute** ***IP_address_of_remote_network_host***.
  b. Examine the results to determine how long the packet took to reach each network segment and the point where the connection can stop working.

Table 44: TCP/IP Diagnostic Utilities

Utility	Used to...
show arp	View the Address Resolution Protocol (ARP) table on the local computer to detect invalid entries.
show hosts	View the hostname, DNS configuration, and static host mappings.
show ip {default-gateway | route}	View current TCP/IP network configuration values.
ping	Verify whether TCP/IP is configured correctly and that a remote TCP/IP system is available.
traceroute	Check the route to a remote system.
show interface <eth number>	check whether errors are reported on the interface.

Related Documentation

- Configuring Interfaces, Hostname, Domain List, DNS, and Default Gateway (CLI) on page 28
- Setting Network Connection Options (CLI) on page 52

Testing Media Flow Controller Delivery Functions

Problem The Wget test (an open source Web crawler program) and Curl test demonstrates how Media Flow Controller uses a configured namespace to fetch and deliver content using HTTP or NFS, as well as Media Flow Controller caching mechanisms.

Solution To perform this test you need a client machine with Wget installed that can also serve as the origin server, and a Media Flow Controller, and connectivity between the two of

them. The setup for this example includes creating data files (**test.txt**) and placing them in a directory (**testresults/joe**) on a UNIX machine with Wget installed, serving as client and origin (**172.16.254.1** / **sv05**).

The actions for this example are configuring a namespace (**testHttp**), requesting files using the Media Flow Controller (**172.16.254.2** / **MFC**), and observing results using **show counters** and the access log on the Media Flow Controller Web interface.

Related Documentation

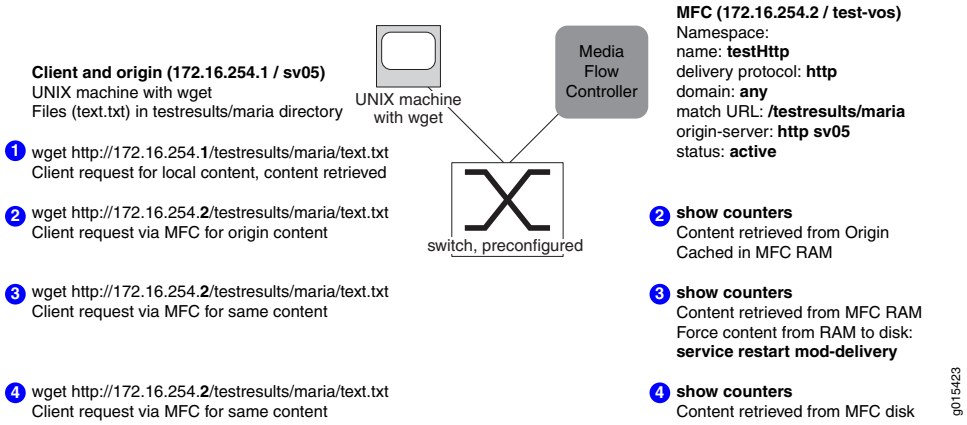
- Policy Engine Overview on page 245
- Configuring NFS Fetch for Images (CLI) on page 144
- Configuring HTTP Fetch for Videos (CLI) on page 144

Testing HTTP Origin Fetch

Problem Test the Media Flow Controller HTTP origin fetch function.

Solution Prepare for the test by doing the following, then follow the steps as illustrated in Figure 26 on page 429:

Figure 26: wget Test for Media Flow Controller HTTP Delivery and Cache



1. Log in to the client/origin UNIX machine and go to a test directory; for example, **testresults/joe**; create a simple text file, **test.txt**.
2. Log in to the Media Flow Controller and configure a namespace, **testHttp**; specify a **uri-prefix** with a domain, delivery protocol, and origin server; and make the namespace active. Example:

```
MFC (config) # namespace testHttp
MFC (config namespace testHttp) # delivery protocol http
MFC (config namespace testHttp) # domain any
MFC (config namespace testHttp) # match uri /testresults/joe
MFC (config namespace testHttp) # origin-server http sv05
MFC (config namespace testHttp) # status active
MFC (config namespace testHttp) # exit
```

To test HTTP origin fetch:

1. From the client/origin machine, use **wget** to fetch the file locally (verify Wget). Example:

```
[joe@sv05 joe]$ wget http://172.16.254.1/testresults/joe/test.txt
--13:12:58-- http://172.16.254.1/testresults/joe/test.txt
Connecting to 172.16.254.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 165 [text/plain]
Saving to: `test.txt.2'
100%[======================================>] 165 --.-K/s in 0s
13:12:58 (15.7 MB/s) - `test.txt.2' saved [165/165]
```

2. Use **wget** to fetch the file using Media Flow Controller. When Media Flow Controller receives the first request for that namespace, it begins logging. Media Flow Controller receives the request, matches the **uri-prefix** to the namespace, and uses the origin server defined for that namespace to retrieve the content. Use **show counters** on the Media Flow Controller to see what happened.
   Example (output truncated):

```
[joe@sv05 joe]$ wget http://172.16.254.2/testresults/joe/test.txt
--13:18:00-- http://172.16.254.2/testresults/joe/test.txt
Connecting to 172.16.254.2:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 165 [text/plain]
Saving to: `test.txt.3'
100%[======================================>] 165 --.-K/s in 0s
13:18:00 (26.2 MB/s) - `test.txt.3' saved [165/165]
[joe@sv05 joe]$
MFC (config) # show counters
Total number of Active Connections : 0
Total Bytes served from RAM cache : 0 Bytes
Total Bytes served from Origin Server: 451 Bytes
Total Bytes served from HTTP Origin Server : 451 Bytes
Total Bytes served from NFS Origin Server : 0 Bytes
Total Bytes served from Disk cache : 0 Bytes
Total Bytes served : 451 Bytes
Total number of HTTP Connections : 1
Total number of HTTP Transactions : 1
Total number of HTTP 200 responses : 1
Total HTTP Well finished count : 1
MFC (config) #
```

3. Run the test again to observe Media Flow Controller serve the content from RAM. Example (output truncated):

```
[joe@sv05 joe]$ wget http://172.16.254.2/testresults/joe/test.txt
--14:07:21-- http://172.16.254.2/testresults/joe/test.txt
Connecting to 172.16.254.2:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 165 [text/plain]
Saving to: `test.txt.4'
100%[======================================>] 165 --.-K/s in
0.002s
14:07:21 (83.4 KB/s) - `test.txt.4' saved [165/165]
[joe@sv05 joe]$
MFC (config) # show counters
Total number of Active Connections : 0
Total Bytes served from RAM cache : 461 Bytes
Total Bytes served from Origin Server : 451 Bytes
Total Bytes served from HTTP Origin Server : 451 Bytes
Total Bytes served from NFS Origin Server : 0 Bytes
```



```
Total Bytes served from Disk cache : 0 Bytes
Total Bytes served : 912 Bytes
Total number of HTTP Connections : 2
Total number of HTTP Transactions : 2
Total number of HTTP 200 responses : 2
Total HTTP Well finished count : 2
```

4. Restart the delivery service so everything in RAM is moved to disk, then run the test once more to observe Media Flow Controller serve the content from disk. Example:

```
MFC (config) # service restart mod-delivery
[joe@sv05 joe]$ wget http://172.16.254.2/testresults/joe/test.txt
--16:17:55-- http://172.16.254.2/testresults/joe/test.txt
Connecting to 172.16.254.2:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 165 [text/plain]
Saving to: `test.txt.3'
100%[======================================>] 165 --.-K/s in 0s
16:17:55 (13.1 MB/s) - `test.txt.3' saved [165/165]
[joe@sv05 joe]$
MFC-cl11 (config) # show counters
Total number of Active Connections : 0
Total Bytes served from RAM cache : 0 Bytes
Total Bytes served from Origin Server : 0 Bytes
Total Bytes served from HTTP Origin Server : 0 Bytes
Total Bytes served from NFS Origin Server : 0 Bytes
Total Bytes served from Disk cache : 461 Bytes
Total Bytes served : 461 Bytes
Total number of HTTP Connections : 1
Total number of HTTP Transactions : 1
Total number of HTTP 200 responses : 1
Total HTTP Well finished count : 1
```

Related Documentation

- Policy Engine Overview on page 245
- Configuring HTTP Fetch for Videos (CLI) on page 144

Testing NFS Origin Fetch

Problem Test the Media Flow Controller NFS origin fetch function.

NFS origin fetch is very similar to HTTP origin fetch, but the namespace configuration differs slightly. NFS has much more functionality than HTTP. When you configure the namespace, you give the URI origin server an NFS IP address (or hostname) and full path—the **uri-prefix** can be anything (for example **nfs1**)—and NFS automatically creates that directory when the first request comes in. The request must include the configured **uri-prefix**.

Solution Prepare for the test by doing the following, then follow the steps as illustrated in Figure 26 on page 429 (note the **wget** path change for the NFS test):

1. Log in to the client/origin machine and go to a test directory; for example, **testresults/joe**; create a simple text file, **test.txt**; and add content to give the file some weight.

2. On the Media Flow Controller, create a new namespace, **testNfs**, and specify a uri-prefix with a domain, delivery protocol, and origin server; then make the namespace active. Example:

```
MFC (config) # namespace testNfs
MFC (config namespace testNfs) # domain any
MFC (config namespace testNfs) # match uri /nfs1
MFC (config namespace testNfs) # origin-server nfs sv05:home/joe
MFC (config namespace testNfs) # status active
MFC (config namespace testNfs) # exit
```

To test NFS origin fetch:

1. From the client/origin machine, use **wget** to fetch the file locally (verify Wget). Example:

```
[joe@sv05 joe]$ wget http://172.16.254.1/testresults/joe/test.txt
--13:12:58-- http://172.16.254.1/testresults/joe/test.txt
Connecting to 172.16.254.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 165 [text/plain]
Saving to: `test.txt.2'
100%[======================================>] 165 --.-K/s in 0s
13:12:58 (15.7 MB/s) - `test.txt.2' saved [165/165]
```

2. From the client/origin machine, use **wget** to fetch the file using Media Flow Controller. When Media Flow Controller receives the first request for that namespace, it begins logging. Media Flow Controller receives the request, matches the **uri-prefix** to the namespace, and uses the origin server defined for that namespace to retrieve the content. Use **show counters** on the Media Flow Controller to see what happened. Example (output truncated):

```
[joe@sv05 joe]$ wget -O newtest http://172.16.254.2/nfs1/test.txt --
17:34:26-- http://172.16.254.2/nfs1/test.txt
Connecting to 172.16.254.2:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 165 [text/html]
Saving to: `newtest'
100%[======================================>] 165 --.-K/s in 0s
17:34:26 (26.2 MB/s) - `newtest' saved [165/165]
[joe@sv05 joe]$
MFC (config) # show counters
Total number of Active Connections : 0
Total Bytes served from RAM cache : 0 Bytes
Total Bytes served from Origin Server : 451 Bytes
Total Bytes served from HTTP Origin Server : 0 Bytes
Total Bytes served from NFS Origin Server : 451 Bytes
Total Bytes served from Disk cache : 0 Bytes
Total Bytes served : 451 Bytes
Total number of HTTP Connections : 1
Total number of HTTP Transactions : 1
Total number of HTTP 200 responses : 1
Total HTTP Well finished count : 1
MFC (config) #
```

3. Run the test again to observe Media Flow Controller serve the content from RAM. Example (output truncated):

```
[joe@sv05 joe]$ wget -O newtest http://172.16.254.2/nfs1/test.txt
--14:07:21-- http://172.16.254.2/nfs1/test.txt
```

```
Connecting to 172.16.254.2:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 165 [text/plain]
Saving to: `newtest'
100%[======================================>] 165 --.-K/s in
0.002s
14:07:21 (83.4 KB/s) - `newtest' saved [165/165]
[joe@sv05 joe]$
MFC (config) # show counters
Total number of Active Connections : 0
Total Bytes served from RAM cache : 461 Bytes
Total Bytes served from Origin Server : 451 Bytes
Total Bytes served from HTTP Origin Server : 0 Bytes
Total Bytes served from NFS Origin Server : 451 Bytes
Total Bytes served from Disk cache : 0 Bytes
Total Bytes served : 912 Bytes
Total number of HTTP Connections : 2
Total number of HTTP Transactions : 2
Total number of HTTP 200 responses : 2
Total HTTP Well finished count : 2
MFC (config) #
```

4. Restart the delivery service so everything in RAM is moved to disk, then run the test once more to observe Media Flow Controller serve the content from disk. Example:

```
MFC (config) # service restart mod-delivery
[joe@sv05 joe]$ wget -O newtest http://172.16.254.2/nfs1/test.txt
--16:17:55-- http://172.16.254.2/nfs1/test.txt
Connecting to 172.16.254.2:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 165 [text/plain]
Saving to: `newtest'
100%[======================================>] 165 --.-K/s in 0s
16:17:55 (13.1 MB/s) - `newtest' saved [165/165]
[joe@sv05 joe]$
MFC-cl11 (config) # show counters
Total number of Active Connections : 0
Total Bytes served from RAM cache : 0 Bytes
Total Bytes served from Origin Server : 0 Bytes
Total Bytes served from HTTP Origin Server : 0 Bytes
Total Bytes served from NFS Origin Server : 0 Bytes
Total Bytes served from Disk cache : 461 Bytes
Total Bytes served : 461 Bytes
Total number of HTTP Connections : 1
Total number of HTTP Transactions : 1
Total number of HTTP 200 responses : 1
Total HTTP Well finished count : 1
```

Related Documentation

- Configuring NFS Fetch for Images (CLI) on page 144
- Configuring HTTP Fetch for Videos (CLI) on page 144

Troubleshooting Layer 3 Forwarding

If the core software component (Caching Engine) of the Media Flow Controller fails, the Layer 3 Forwarding Enable feature routes new TCP connections directly to the origin servers. Layer 3 Forwarding prevents black-holing TCP connections destined for TCP

Port 80 (HTTP protocol) if the Caching Engine has an outage. If you enable Layer 3 Forwarding, Media Flow Controller behaves as follows:

- Existing connections are terminated
- New connections are forwarded through the Layer 3 Forwarding feature to the Origin Server.
- Alarms are raised to denote the outage of the Caching Engine.
- If the Caching Engine has an outage due to a software error or crash, Media Flow Controller writes a core file for Juniper Networks to debug the problem. You must configure this feature separately.
- When the Caching Engine restarts, it terminates new connections normally.

NOTE: The core dump contains technical support information, such as the system information, core file, and debug information at the time of the crash). Contact Juniper Networks technical support with the core dump file for troubleshooting purposes. Typically, the core dump file is generated at the **/coredumps/snapshots/** location as a ***.tar.gz** file.

To enable Layer 3 forwarding:

1. Enter the CLI configure terminal mode.

   > **enable**

   # **configure terminal**

2. Enable a bypass when the cache delivery engine fails.

   # **service bypass-on-failure mod-delivery enable**

NOTE: Before entering the **service bypass-on-failure mod-delivery enable** command, ensure that you have entered the **network connection ip-forward** command.

To disable Layer 3 forwarding:

1. Enter the CLI configure terminal mode.

   > **enable**

   # **configure terminal**

2. Disable a bypass when the cache delivery engine fails.

   # **service bypass-on-failure mod-delivery disable**

Related Documentation

- Before You Configure Media Flow Controller on page 20

Testing a Specific Transaction

Problem A specific video is having delivery problems.

Solution Use the trace log facility to debug.

Media Flow Controller includes a trace facility to help diagnose the handling of a particular HTTP request. The tracing is done by logging trace points, written to the trace log. A request is traced if the following conditions are true:

- The global trace flag is enabled using the CLI:

  delivery protocol http trace enable

- The HTTP request includes the **X-NKN-Trace** header; for example, if using Wget:

  wget --header "X-NKN-Trace:" *URL*

The HTTP modules detect the above conditions and set the flag " HRF_TRACE_REQUEST," as well as other flags that direct each relevant module to log meaningful trace points.

To trace the path of a specific video:

1. Set the delivery trace.

   delivery protocol http trace enable

2. Use Wget (or other tool such as cURL) to request the problem video with this header added:

   X-NKN-Trace

To see the trace points and for additional information, see "Reading the Trace Log" on page 347.

Related Documentation

- Reading the Trace Log on page 347

Enabling Debug Operations

Media Flow Controller provides a system-level debugging utility as well as a trace facility; see "Reading the Trace Log" on page 347 for details.

CAUTION: The system gives high priority to debugging output. For this reason, debugging commands should be turned on only for troubleshooting specific problems or during troubleshooting sessions with technical support personnel. Excessive debugging output can render the system inoperable.

To enable debugging:

1. Generate debugging information for all system functions.

   debug generate dump

2. View a list of debug-dump files.

   show files debug-dump

3. View a summary of a specific debug-dump file.

   show files debug-dump *filename_of_debug_dump*

4. E-mail a generated debug-dump file to a list of recipients configured to receive **info** event notices (see "Configuring Media Flow Controller Fault Notifications (CLI)" on page 372 for task details).

   file debug-dump email *filename_of_debug_dump*

5. Upload a specific debug-dump file. The uploaded file is a gnu-zipped .tar file (.tgz) and can be unzipped with the **gunzip -c *filename*.tgz | tar tf -** command on Linux, or with WinZip on a Windows system.

   file debug-dump upload *filename_of_debug_dump URL*

NOTE: Only FTP and TFTP URLs, as well as SCP pseudo-URLs, are supported for the destination.

Related Documentation

- Reading the Trace Log on page 347
- Configuring Media Flow Controller Fault Notifications (CLI) on page 372

Viewing Media Flow Controller License Information

You can view the Media Flow Controller license information from the CLI or web interface.

If you are using the CLI to view the license information, you can run the **show licenses** command to view the currently installed licenses and their validity. Some of the scenarios and the corresponding status messages are listed below:

- If no licenses are installed:

  mfc-unconfigured-bc9606 (config) # show licenses

  No licenses have been configured.

- If one of the licenses has expired:

  mfc-unconfigured-bc9606 (config) # show licenses

  License 1:
  LK2-MFD-44GT-1G68-V30C-1H68-Q32C-1G60-Q30C-1N81-GT35-DJM8-4JDN-Y4PA-MLE2-D17U-WU25
  Feature: MFD
  Valid: yes
  End date: 2012/02/01 (FAILED)
  Tied to host ID: 0280012010000006 (ok)
  Active: no

If you are using the web interface, you can view the license information on the top right-hand corner of the web interface.

- If the license has expired, then you may see:
  License : Product disabled. License expired on : <*date-of-expiry*>.
- If the license is malformed, then you may see:
  License : Product disabled. Malformed license.

Related Documentation

- Troubleshooting Media Flow Controller Invalid Licenses on page 437

Troubleshooting Media Flow Controller Invalid Licenses

Problem Media Flow Controller licenses can be invalid. You might have older licenses that used a different scheme.

Solution Media Flow Controller licenses can tied either to the Host ID or the MAC address of Eth0. Typically you must manually set the Eth0 interface with one of the **Ethernet naming** options during installation. If not specified, the default assignment of ports is the order in which the drivers are loaded. If Eth0 is not assigned to the MAC address associated with your licenses at that time, your licenses will be invalid. If this happens, you should reinstall Media Flow Controller and use the **Ethernet naming** options to name the correct interface Eth0. However, if reinstallation is not an option, you can use the management interface command to name the correct interface as Eth0.

To check the Host ID and licenses, use the following management interface commands:

1. Determine the Host ID to which your licenses are tied.

   show license

2. Determine the version to which your Host ID is tied.

   show version

3. Compare the Host ID between license and the version. If they are different, the license is invalid or the license has expired.

Related Documentation

- Applying the Media Flow Controller License (CLI) on page 48
- Installing Licenses (Web Interface) on page 318

Troubleshooting the namespace match uri Configuration

Problem Requests are being misdirected. You might have configured the namespace **match uri *uri-prefix*** incorrectly or you might need to set **precedence**.

Solution Proper namespace configuration is required for smooth Media Flow Controller functioning. Namespace is how Media Flow Controller knows what to deliver and from where to fetch it, if needed. An important element is the **match** criteria for the namespace. A **match uri *uri-prefix*** can be very specific; for example, **/vod/path1/path2**; or very unspecific, for

example, / (slash). An unspecific ***uri-prefix*** can be thought of as a superset. In the case of a ***uri-prefix*** of a / (slash), all the requests will map to that namespace because all the requests will have a / (slash) in them. If you want a superset ***uri-prefix***, it is important to also set a precedence for that namespace so that Media Flow Controller checks at other namespaces first. For more information, see "Using namespace precedence" on page 96.

Related Documentation

- Using namespace precedence on page 96

Troubleshooting namespace domain Configuration

Problem Incoming requests are being rejected. You might have configured the **namespace domain** incorrectly.

Solution When configuring **namespace domain**, make sure the domain you enter matches whatever you expect as the Host header in requests. You can also append a port number, if needed (and used in the Host header). If you append a port number that is not the default and is not in the Host header, the request will fail. See "Using namespace domain <FQDN:port>" on page 96 for more details.

Related Documentation

- Using namespace domain <FQDN:port> on page 96

Troubleshooting File Not Getting Cached

Problem The file fetched from the origin is not getting cached in disk.

Solution Refer Table 45 on page 438 for tunnel codes when you observe a file being fetched from the origin but not being cached in the disk:

Table 45: Tunnel Codes

Tunnel Code	Definition
0	Unknown reasons, validate failed or other complicated paths
1	It is a bad request from client which can be tunneled
2	Origin selection method is Abs URL, but request is not cacheable
3	There is no Host header
4	PE set to do not cache
5	SSP set to force tunnel
6	Request COD returned NULL (hostname>128, port_len>5, err creating COD)
7	URI size greater than 512 bytes

Table 45: Tunnel Codes *(continued)*

8	Neither GET nor HEAD requests
9	Accept content-type does not match the request
10	Force tunnel for all requests
11	Request has content length and not zero
12	Multi-byte mime header exists in request
13	Some data exists in addition to http header
14	Authorization header present
15	Cookie header present
16	Request has query parameters
17	Request has cache-control set to no-cache
18	Request has pragma set to no-cache
19	Request has transfer encoding chunked
20	Host exists but its not valid FQDN
21	Host in absolute URI is not valid FQDN
22	URI does not exist in the request
23	Byte range request, negative
24	HTTP version not 1.0 or 1.1
25	HTTP version 1.1 without Host header
26	Transfer type encoding request
27	Byte range request, stop range < start range
28	Dynamic uri regex match fail
29	Client request size is more than configured
30	Request encoding type is not supported
31	Request T-proxy incapable byte range origin tunnel
65	PE set to do not cache

Table 45: Tunnel Codes *(continued)*

66	URI with query string and disable cache on query is configured
67	Both transfer-encoding: byte-range and transfer_encoding: chunked exist
68	Neither content_length nor transfer_encoding: chunked exist
69	content_length exists but it is zero
70	URI has unprintable characters and virtual player is not configured
71	Server returns 404 and SSP is not configured
72	Unsupported response from the server (not 200, 404, 100 etc.)
73	Handle 302 is not configured and its not reverse proxy
74	302 response without location header
75	302 response and location header has errors
76	COD verification failed
77	Cache control set to no transform
78	Failed to save request state for validate
79	Failed to save mime header
80	Failed to serialize ocon mime_header_t into attribute buffer
81	Error in getting content range from response
82	Response has invalid content range data
83	No content length
84	Received OS data offset not same as request offset
85	Response has Cache control set to private
86	Response has set-cookie
87	Object already expired, caching disabled
88	Adding query to attributes failed
89	Cap of Cache Object Size
90	Non cacheable, because it has some ache control directive

Table 45: Tunnel Codes *(continued)*

91	Object not found in cache
92	Chunked response, but cache-age is less than what's configured
93	Response encoding type is not supported
94	Cache index based response has no configured header value in reply
95	Cache index based response URI exceeds maximum URI size
96	Cache index based response has out of range data bytes
97	Dynamic URI tokenization error
98	Cache index response data configured for chunked object
99	HTTP 0.9 requests
100	WWW-Authentication: NTLM needs connection level tunnel
101	Internal crawl request
102	Server response size is more than the configured one
103	Chunked request which is sub-encoded with some other format
104	If ETag/Last-Modified Header (LMH) not there and COD version ignore not set
105	Response R-proxy incapable byte range origin tunnel

Related Documentation
- Configuring Caching All Contents for a Website (CLI) on page 72

Troubleshooting Cache Promotion Not Happening

Problem Objects in cache are not getting promoted correctly. You might have disabled cache promotion while debugging.

Solution
- Verify that cache promotion has not been disabled. Use the **show analytics** command.
- If cache promotion has been disabled, enable it with the **analytics cache-promotion enable** command.

Related Documentation
- Configuring Caching All Contents for a Website (CLI) on page 72

Troubleshooting Incoming Requests' URL Length

Problem Incoming requests are being rejected. The system throws tunnel code 7. The URI length might have exceeded the maximum allowed length.

Solution The default acceptable URL length (domain + URI + Query Params + Headers), in characters or bytes, for incoming requests is **16384** bytes; the maximum allowed value is **32768**. Incoming requests with lengths exceeding the set value are rejected. You can modify this with the **delivery protocol** argument **req-size incoming maximum**. See **delivery** in the *Media Flow Controller CLI Command Reference* for CLI details.

There are two thresholds:

- Max URL length—This is the length of the URL itself. A URL consists of (domain + [URI + query params]). The size of a URI cannot exceed 512 bytes. Any URI beyond this size is treated as non-cacheable. There is no CLI to configure this limit at present.
- Max Request length—This is the total size of a request which, as described, must not exceed the configured size.

Related Documentation
- Policy Examples Based on pe_http_recv_request on page 269
- Policy Engine API Reference on page 257

Troubleshooting Accesslog SFTP

Problem There is a problem sending access log using SFTP.

Solution To configure the access log copy for SFTP:

(config) # accesslog copy *sftp://user@host:path*

To set up the SSH keys:

1. Add the RSA key from the target host to which you want to push the log (for example **/etc/ssh/ssh_host_rsa_key.pub**) to Media Flow Controller.

   (config) # ssh client global known-host "*IP* ssh-rsa ..."

2. Generate the public/private keys in Media Flow Controller.

   (config) # ssh client user admin identify rsa2 generate

3. Take the Media Flow Controller's public key and put in the target host's authorized keys.

   (config) # show ssh client

   For example, take the public key (Media Flow Controller):

   ```
   User admin:
   RSAv2 Public key:
   ssh-rsa ....
   ```

   And place it here (client machine); for example **/user_name/.ssh/authorized_keys**.

4. To verify, do a manual upload.

   (config) # upload accesslog current sftp://user@host:path (i.e. sftp://root@192.168.1.10:/tmp)

The first time you do the manual upload, you'll see something like this:

```
(config) # upload accesslog current sftp://root@192.168.1.10:/tmp
sftp> cd /tmp
sftp> put access2.log access2.log.tmp
Uploading access2.log to /tmp/access2.log.tmp
sftp> -rm access2.log
Couldn't stat remote file: No such file or directory Removing /tmp/ access2.log Couldn't
   delete file: No such file or directory
sftp> rename access2.log.tmp access2.log exit
(config) #
```

Related Documentation

- Configuring Media Flow Controller Service Logs Overview on page 333

Troubleshooting a Lost Admin Password

Troubleshooting a Lost Admin Password

Problem The administrator password has been lost.

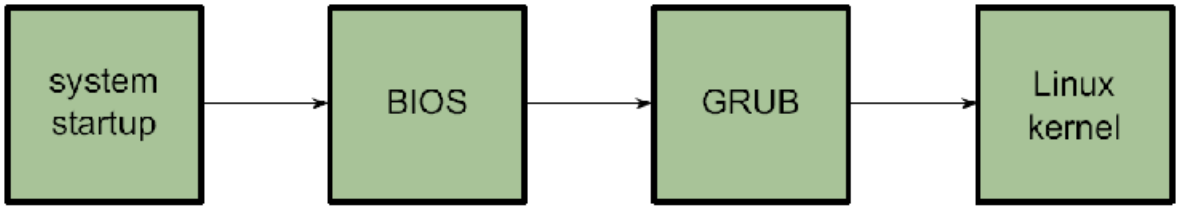
Solution The procedure for resetting a lost admin password for Media Flow Controller is based on the conventional procedure for resetting a lost Linux password, with a few important differences. The most important difference is that Media Flow Controller uses a database to store passwords instead of the standard Linux **/etc/passwd** file. For that reason, the procedure described below uses a shell script to reset the password database.

NOTE: **The procedures described in this section assume broad familiarity with Linux system administration.**

About the Media Flow Controller Boot Process

Media Flow Controller uses a standard Linux boot process, shown in Figure 27 on page 444. To reset the admin password, you must interrupt the boot process and force Media Flow Controller into Linux single-user mode. You then run a shell script to reset the password database.

Figure 27: Media Flow Controller Boot Process



The procedures below are based on the GRUB bootloader and the modifying the Linux kernel command line.

NOTE: This procedure assumes that the boot media priority has not been altered from the system default (hard drive).

Resetting the Password Database

To reset the password database for Media Flow Controller:

1. Force a reboot if the CLI is still running:

   a. Enter enable mode.
   MFC> **enable**

   b. Reboot Media Flow Controller.
   MFC> **reload**

   If the CLI is not running, then you probably cannot log in to the **admin** account. In this case, you must perform a hardware reboot.

2. Enter the GRUB bootloader command-line interface by pressing and releasing the Space key during the initial boot sequence after GRUB begins to display a series of periods (**...**).

   The following GRUB bootloader menu appears, describing the two operating system images that can be booted:

```
----------------------------------------------------------------------
0: mfc-12.1.0 180_13291_229
1: mfc-12.1.0 180_13291_229
----------------------------------------------------------------------
```

3. Use the arrow keys to select an operating system image. In most cases, you can see the first image. Press **e**.

   You see a series of operating system kernel command lines:

```
----------------------------------------------------------------------
0: root (hd0,1)
1: kernel /vmlinuz ro root=LABEL=ROOT_1 rootdelay=5 img_id=1 quiet
loglevel=4 panic=10 console=tty0 console=ttyS0,9600n8
2: initrd /initrd.img
----------------------------------------------------------------------
```

4. Use the arrow keys to select line 1. Press **e** to begin editing the kernel command-line options.

   You see following kernel command line for editing:

   ```
   Highlighted entry is 1:
   [ Minimal BASH-like line editing is supported. For the first word,
   TAB lists possible command completions. Anywhere else TAB lists
   the possible completions of a device/filename. ESC at any time
   cancels. ENTER at any time accepts your changes.]
   <1 quiet loglevel=4 panic=10 console=tty0 console=ttyS0,9600n8
   ```

5. Edit the kernel command line.

   a. Use the arrow keys to select line 1. Press the down arrow key once. The message at the bottom should now read:

      Highlighted entry is 1:

   b. Append the word **single** to the end of the command line.

      This change is temporary and is reset automatically at the next reboot.

   c. Press **Enter** to complete the changes to the command line.

   d. Press **b** to complete the boot sequence into single-user mode.

6. Reset the password database:

   NOTE: Typical Linux systems store their passwords in the **/etc/passwd** file. Media Flow Controller uses a database to maintain passwords. The password reset script **/sbin/resetpw.sh** performs all the necessary changes to this database.

   a. Run the password reset script.

      # /sbin/resetpw.sh

      If successful, the **admin** password is reset to the empty password. If any corruption in the configuration state is detected, an attempt to save existing configuration databases is made and the system's configuration state is reset to its initial values (the same state just after manufacturing the system). In this case, the **admin** password is set to an empty password. The command will notify the user of any locations of the saved configuration state.

   b. Reboot the system.

      # reboot

      The previous changes made to the kernel command line are reset automatically and the system reboots normally with a reset password database.

Related Documentation

- Configuring Media Flow Controller User Accounts (CLI) on page 47

Troubleshooting No Web Interface Access

Problem The Web interface does not connect.

Solution Access to the Web interface relies on several factors:

- You must have an IP address set for the management interface, typically Eth0, although this is not a requirement.
- You must have a default gateway (Internet or network access) set, unless you are plugged directly into the console.
- The IP address you enter on the Location bar of your browser must specify port 8080 (add ":8080" after the IP address).
- The Web interface must be enabled (**web enable**), enabled to listen (**web httpd listen enable**), and assigned to Eth0 (**web httpd listen interface eth0**). These are all default settings, but could need resetting if the Web interface is not connecting.

If all the above criteria are met and the problem still exists, restart the httpd process.

```
(config) # pm process httpd restart
```

Related Documentation

- Logging In to Media Flow Controller for the First Time (Web Interface) on page 283

CHAPTER 15

Media Flow Controller Deployment Guidelines

Media Flow Controller Deployments Overview

Media Flow Controller can be deployed by content publishers, Internet service providers, and content delivery networks (CDNs) for origin acceleration and edge caching as a reverse proxy, transparent proxy or mid-tier proxy.

Table 46: Transparent Proxy versus Reverse Proxy

Transparent Proxy	Reverse Proxy
Requests redirected to the cache using policy-based routing.	Requests redirected to the cache using DNS-based routing.
Typically deployed at Internet service providers (ISPs) or campus edges only.	Typically deployed at origin, content publisher, or CDN edges. Can also be deployed at ISP or campus edges.
Requires policy changes at all the edge routers handling user requests.	Requires changes in the DNS server.
Does not require contracts with the content publisher.	May require contracts with the content publisher.
Server and client do not know there is a cache in-between.	Server and client know there is a cache in-between.

Related Documentation

- Reverse Proxy Deployments on page 448
- Reverse Proxy Deployment Requirements on page 449
- Reverse Proxy Deployment Process on page 450
- Reverse Proxy Cache Tuning CLI Commands on page 452

Reverse Proxy Deployments

Reverse Proxy Deployments

About Reverse Proxies

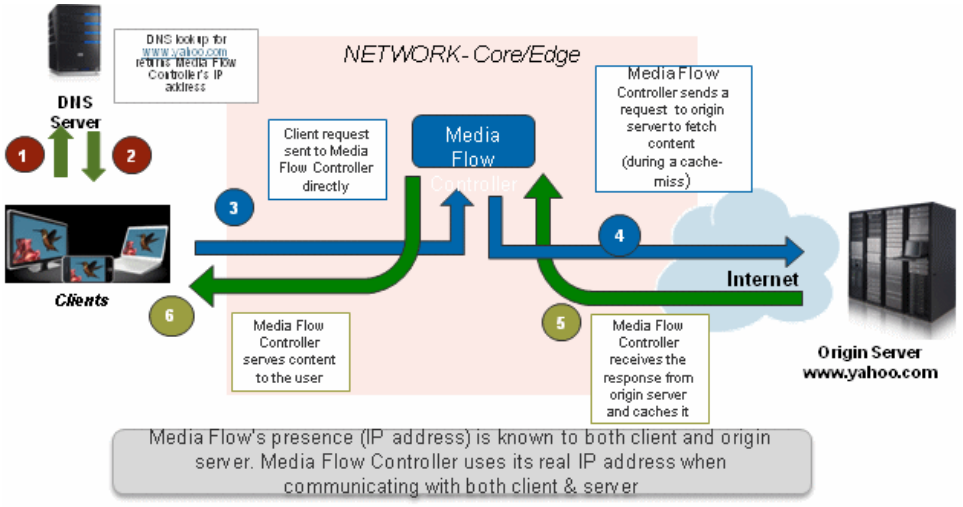
In a reverse proxy deployment, the user's request is routed to Media Flow Controller using DNS-based redirection or by using an SLB/GSLB module or by using HTTP redirects.

Deploy Media Flow Controller as a reverse proxy when one or more of the following criteria are met:

- You want to do caching at the origin or edge
- You are a Content Publisher, CDN or an ISP/Campus Edge operator
- You do not want to change policies or configuration in the routers/switches
- You are ready to change DNS configuration
- You are fine with the user and origin knowing about the presence of a proxy/cache in-between

See Figure 28 on page 449 for a graphic depicting a reverse proxy call flow.

Figure 28: Reverse Proxy Call Flow



Reverse Proxy Protocol Support

Media Flow Controller supports content acquisition or publishing using HTTP, NFS, FTP, RTP/RTSP, and RTMP protocols. Media Flow Controller delivers media to users using the HTTP, RTP/RTSP, and RTMP protocols. Supported combination of protocols for media fetch and delivery are given in "Media Flow Controller Minimum System Requirements" on page 4.

Related Documentation

- Media Flow Controller Deployments Overview on page 447
- Reverse Proxy Deployment Requirements on page 449
- Reverse Proxy Deployment Process on page 450
- Reverse Proxy Cache Tuning CLI Commands on page 452

Reverse Proxy Protocol Support

Media Flow Controller supports content acquisition or publishing using HTTP, NFS, FTP, RTP/RTSP, and RTMP protocols. Media Flow Controller delivers media to users using the HTTP, RTP/RTSP, and RTMP protocols. Supported combination of protocols for media fetch and delivery are given in "Media Flow Controller Minimum System Requirements" on page 4.

Related Documentation

- Media Flow Controller Deployments Overview on page 447
- Reverse Proxy Deployment Requirements on page 449
- Reverse Proxy Deployment Process on page 450
- Reverse Proxy Cache Tuning CLI Commands on page 452

Reverse Proxy Deployment Requirements

You need the following information to get started with the configuration:

- **Type of delivery**—Live or on-demand video delivery, or Web content delivery
- **Delivery protocol**—HTTP, RTP/RTSP, or RTMP
- **Ingest Protocol**—HTTP, NFS, RTP/RTSP, or RTMP
- **Origin server details**—Hostname, IP address
- **Mapping request to origin policies**—The list of parameters in the URL (domain, URI, and query parameters) for caching and delivery policies

Related Documentation

- Media Flow Controller Deployments Overview on page 447
- Reverse Proxy Deployments on page 448
- Reverse Proxy Deployment Process on page 450
- Reverse Proxy Cache Tuning CLI Commands on page 452

Reverse Proxy Deployment Process

We assume that the Media Flow Controller is configured with the basic settings such as license configuration, interface configuration, IP address assignment, hostname, domain list, system clock, DNS, IP routes, and default gateway configuration. Once the basic configurations are done and the required information gathered, you are ready to begin the deployment process, as outlined:

1. Create a namespace—A namespace contains a set of policies that influence how content is fetched from origin server, cached and delivered to users. You can create multiple namespaces if you want to apply separate set of policies for each of the properties, domains, or customers. Examples of reverse proxy namespace configurations for different deployments are given in "Reverse Proxy Namespace Examples" on page 455.
2. Map requests to namespaces by configuring a namespace domain and namespace match criteria—See "Creating a Namespace and Setting Namespace Options (CLI)" on page 76 for details on using namespaces.
3. Define origin server and ingest protocol settings—Configure origin server settings such as server hostname and IP address, and/or port number for origin fetch, for each namespace.
   If you have multiple origin servers and you want to distribute load across different origin servers, or if you want configure origin server failover settings, use the server-map feature in Media Flow Controller. If you have Media Flow Controllers upstream (mid-tier), you can use the Cluster map type server map.

   For information on using the origin-server option, see "Creating a Namespace and Setting Namespace Options (CLI)" on page 76. For information on using the server-map options, see "Configuring Server Maps (CLI)" on page 232

   For IPv6 support, you configure the **namespace absolute-url** option with the IPv6 address to derive the origin IP address from the client request. Verify that the **namespace ip-version follow-client** option is set to select the IP version of the client for connecting to the origin server.

4. Define delivery protocols—Configure the protocol or protocols to deliver media to users; your options are HTTP, RTSP, or RTMP.

   For information on for HTTP or RTSP delivery, see "Configuring Media Flow Controller Delivery Protocols (CLI)" on page 54.

   For IPv6 support, you must enable IPv6 for delivery protocols configuration by using the **delivery protocol http ipv6 enable** command.

5. Define type of delivery—No special configuration is required to enable on-demand delivery, or HTTP-based live delivery.

   You must define a **namespace *name* live-pub-point** to configure settings related to RTSP live streaming. RTMP streaming must be configured using the Adobe Flash Media Server configuration files.

   For information on configuring live-streaming, see"Using Namespace for Live Streaming Delivery Without Caching" on page 108 and "Using Namespace for Live Streaming Delivery with Caching" on page 108.

6. Configure Media Flow Publisher (if applicable)—Configure Media Flow Publishing if you need to deliver on-demand/live videos using Apple HTTP Streaming, Microsoft SmoothStreaming, or Adobe Dynamic HTTP Streaming methods to QuickTime, Silverlight, and Flash players, respectively.

   For information on Media Flow Publisher, see the *Media Flow Publisher Configuration Guide*.

7. Create virtual-players (if applicable)—Virtual Player is the video requests processing engine that performs the following functions:

   - Acts like a media player when fetching content from the origin server
   - Acts like a media server when delivering content to users

   Virtual Players are typically used in a reverse proxy deployment to perform any of the following functions:

   - Validate URLs by generating a URL hash before delivering video
   - Configure query, or video, " seek" related settings
   - Define how Media Flow Controller should uniquely identify objects in cache

   For information on HTTP or RTSP delivery, see "Virtual Players Overview" on page 117.

8. Tune cache configuration (if required)—Media Flow Controller's default configuration is tuned for optimal caching and delivery performance. You can tune the cache configuration to maximize the performance for the given deployment. Refer to the following sections:

   - "Setting Network Connection Options (CLI)" on page 52—For tuning network connection-related parameters such as maximum bandwidth, connection idle timeout, and concurrent sessions.
   - "Configuring Media Flow Controller Delivery Protocols (CLI)" on page 54—For tuning request/response processing-related parameters such as overriding cache options, object size tuning, header insertion/deletion, and enabling connection pooling

Related Documentation

- Configuring Media Flow Controller Delivery Protocols (CLI) on page 54
- Creating a Namespace and Setting Namespace Options (CLI) on page 76
- Reverse Proxy Deployments on page 448
- Virtual Players Overview on page 117
- Configuring Server Maps (CLI) on page 232
- Reverse Proxy Cache Tuning CLI Commands on page 452
- Setting Network Connection Options (CLI) on page 52

Reverse Proxy Cache Tuning CLI Commands

See Table 47 on page 452 for guidelines on reverse proxy cache performance tuning.

Table 47: Proxy Cache Tuning CLI Commands

Config Parameter	Description	Benefit	Guidelines
namespace *name* delivery protocol (http | rtsp) origin-fetch cache-age-default	Set the maximum age of objects when origin did not specify the maximum age of the object. (Default: 8 hours or 28800 secs)	Keeps the content in the cache for a longer duration, thereby improving cache-hit ratio.	Set this configuration when Media Flow Controller is fetching objects from NFS origins or when objects are published using FTP. The higher the cache-age value, the higher the cache-hit ratio will be (because of reduced churn of objects in cache).
namespace *name* delivery protocol http origin-fetch cache-directive no-cache follow	Do not cache the object when the origin server says so; for example, when origin includes Cache-control: no-cache header.	Prevents dynamically generated or protected content from getting cached.	Set this configuration only in namespaces that serve dynamic content or protected content from origin servers.
namespace *name* delivery protocol http origin-fetch cache-fill aggressive	Fetch the full object from the origin, even though the user requested only partial object.	Can improve the cache-hit ratio by fetching the entire object, instead of downloading just one chunk requested by the given user.	Set this configuration in namespaces that serve popular objects that are large (such as videos, install packages, and PDF files).
namespace *name* delivery protocol http origin-fetch content-store media cache-age-threshold	Short-lived objects are only stored in RAM cache. (Default: Objects of age under 60 seconds, stored in RAM.)	Can improve disk performance by minimizing the churn in disk cache.	Keep the value of cache-age-threshold very low, to improve the disk performance.

Table 47: Proxy Cache Tuning CLI Commands *(continued)*

Config Parameter	Description	Benefit	Guidelines
namespace *name* delivery protocol http origin-fetch content-store media object-size	Set the size below which content is tunneled and not cached at all. (Default: 4096, for example, means cache all objects above 4096 bytes)	Because disk throughput performance is limited by seeks for small objects and small objects make a large number of the object count but a small amount of the overall throughput, you maximize disk throughput by limiting disk caching to large objects.	If the objective is to maximize disk throughput, increase the value of this parameter. The side effect of increasing this parameter value is that the subscriber experience is not improved as much as it would be with a smaller value of this parameter. The subscriber experience is better when small objects are served from cache because page-rendering time typically depends on quickly loading small objects. Therefore, keep this value smaller if subscriber experience is important. See more details about the relationship between this parameter and the value of the **cache-tier highest object-size** in the **namespace *name* delivery protocol http origin- fetch content-store media cache-tier highest object-size** command description found in this table.
namespace *name* delivery protocol http origin-fetch content-store media cache-tier highest object-size	Set the size limit below which the device caches objects in the highest disk cache tier. The disk cache hierarchy is as follows, from lowest to highest: SATA, SAS, SSD. If the system has only one type of disk, this command has no effect. If there are two or more types of disks (for example, SATA and SSD, or SATA and SAS) the device caches those objects with a size below the configured **cache-tier highest object-size** value into the highest tier; those objects equal to or larger than the configured size limit are cached in other tiers. (Default: 0)	Can improve disk cache performance by forcing smaller objects immediately into the highest cache tier.	Use this command in conjunction with the **namespace *name* delivery protocol http origin-fetch content-store media object-size** command. You must set the value for the **cache-tier highest object-size** value in this command higher than the value you set for the **content-store media object-size**. This is because the **content-store media object-size** value sets the size below which content will be tunneled and not cached at all, while the **cache-tier highest object-size** value sets the size above which content is allowed to be cached into the highest tier.

Table 47: Proxy Cache Tuning CLI Commands *(continued)*

Config Parameter	Description	Benefit	Guidelines
namespace *name* delivery protocol http origin-request cache-revalidation permit	Performs automatic revalidation of the object in cache when it expires or when the object is close to become expired.	Can improve cache-hit ratio, by just revalidating the expired object instead of fetching the entire object from origin.	Set this configuration to minimize transit bandwidth usage and to improve cache-hit ratio.
namespace *name* delivery protocol http origin-request connect retry-delay	Retry connection with origin server after a delay of given ms. (Default: 100 ms)	Can improve connection management with origin when dealing with slow origin servers.	Increasing the value of retry-delay minimizes the origin server load. Decreasing the value of retry-delay improves the user experience by reducing the wait time for request retries.
namespace *name* delivery protocol http origin-request connect timeout	Timeout the connection request sent to origin, if there is no response from origin. (Default: 100 ms)	Can improve connection management with origin when dealing with slow origin servers.	Increasing the connect timeout minimizes the load on origin server. Lowering the connect timeout improves the user experience by reducing the wait time for requests.
namespace *name* delivery protocol http origin-request host-header inherit incoming-req deny	Do not inherit "Host" header from the client request.	Distribution of load across origin servers, by not inheriting the Host header from client.	Disable this configuration if you have a load balancer to distribute requests across origin servers. Set this configuration only when you want to distribute requests across origins based on " Host" header.
namespace *name* delivery protocol http origin-request read interval-timeout	Timeout the read request if there is no response from origin for the given duration. (Default: 100 ms)	Can improve throughput, by optimizing the network read operations.	Increasing the read timeout minimizes the load on origin server. Lowering the read timeout improves the user experience by reducing the wait time for requests.

Table 47: Proxy Cache Tuning CLI Commands *(continued)*

Config Parameter	Description	Benefit	Guidelines
namespace *name* delivery protocol http origin-request read retry-delay	Retry read from origin server, after a delay of given duration. (Default: 100 ms)	Can improve throughput, by optimizing the network read operations.	Increasing the value of retry-delay minimizes the origin server load. Decreasing the value of retry-delay improves the user experience by reducing the wait time for request retries.
namespace *name* delivery protocol http origin-request x-forwarded-for enable	Include X-forwarded-for header with client IP address, in the request sent to origin.	Tracks the source of the request.	Enable this always in reverse proxy mode.
namespace *name* delivery protocol http client-request cache-control max-age 0	Do not serve the client request with the object from cache.	Allows users to fetch content from origin, bypassing the cache.	Enable this always in reverse proxy mode.
namespace *name* delivery protocol http client-request cookie action cache	Cache objects though the request contains cookie header	Can improve cache-hit ratio, by caching objects with cookies.	Disable this by default in reverse proxy mode. Set this configuration only when you are sure that the same object can be delivered to multiple users, by Media Flow Controller (irrespective of the user cookies).
namespace *name* delivery protocol http client-request query-string action no-cache	Do not cache object if the request URL contains query parameters. Presence of query parameters in the URL indicates that the response is dynamically generated, such as the output of CGI.	Can improve cache-hit ratio, by not caching content generated by CGIs.	Enable this by default in reverse proxy mode. Disable this configuration only when you are sure that responses to requests with query-parameters can be cached.

Related Documentation

- Media Flow Controller Deployments Overview on page 447
- Reverse Proxy Deployments on page 448
- Reverse Proxy Deployment Process on page 450

Reverse Proxy Namespace Examples

These configurations provide example configurations; variables are indicated with italics.

- Example: HTTP In and HTTP Out on page 456
- Example: NFS In and HTTP Out on page 456
- Example: On-Demand RTSP In and RTSP Out on page 457
- Example: Live RTSP In and RTSP Out on page 457

Example: HTTP In and HTTP Out

- Type of delivery—On-demand (Web content delivery)
- Delivery protocol—HTTP
- Ingest protocol—HTTP
- Origin server details—web.example.com
- Mapping request to origin policies based on domain (example.com) and URI pattern (/web)
  - Namespace configuration:

```
namespace example

    domain example.com
    match uri /web
    origin server http web.example.com
    delivery protocol http
    status active
```

Example: NFS In and HTTP Out

- Type of delivery—On-demand (Web content delivery)
- Delivery protocol—HTTP
- Ingest protocol—NFS
- Origin server details—web.example.com
- Mapping request to origin policies based on domain (example.com) and URI pattern (/web)
  - Namespace configuration:

```
namespace example

    domain example.com
    match uri /web
    origin server nfs nfs.example.com:/var/www/
    delivery protocol http
    status active
```

Example: On-Demand RTSP In and RTSP Out

- Type of delivery—On-demand (video stream delivery)
- Delivery protocol—RTSP
- Ingest protocol—RTSP
- Origin server details—rtsp.example.com
- Mapping request to origin policies based on domain (example.com) and URI pattern (/video)
  - Namespace configuration:

```
namespace example

  domain example.com
  match uri /video
  origin server rtsp rtsp.example.com rtp-rtsp
  delivery protocol rtsp
  status active
```

Example: Live RTSP In and RTSP Out

- Type of delivery—Live (video stream delivery)
- Delivery protocol—RTSP
- Ingest protocol—RTSP
- Origin server details—rtsp.example.com
- Mapping request to origin policies based on domain (example.com) and URI pattern (/livepub)
  - Namespace configuration:

```
namespace example

  domain example.com
  match uri /livepub
  origin server rtsp rtsp.example.com rtp-rtsp
  delivery protocol rtsp
  status active
  live-pub-point cusid1-office-allhands
    receive-mode on-demand
  status active
```

Example: Distribute Requests to Different Origin Servers Based on Host Header

- Type of delivery—On-demand (Web content delivery)
- Delivery protocol—HTTP
- Ingest protocol—HTTP
- Origin server details—host1.example.com, host2.example.com, host3.example.com
- Mapping request to origin policies based on domain (example.com) and URI pattern

(/web)—Host header in the HTTP request contains values such as web.example.com, images.example.com, and video.example.com

- Namespace configuration:

```
namespace example
    domain example.com
    match uri /web
    origin server http server-map OriginServerDefinitions
    delivery protocol
    http status active
```

- Server map configuration:

```
server-map OriginServerDefinitions
    format-type host-origin-map
    file-url http://example.com/serverdefinitions.xml
```

Contents of serverdefinitions.xml:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE HostOriginMap SYSTEM "HostOriginMap.dtd">
<HostOriginMap>
  <Header>
  <Version>1.0</Version>
  <Application>MapXML</Application>
  </Header>
  <HostOriginEntry>
  <Host>web.example.com</Host>
  <Origin>host1.example.com</Origin>
  <Port>80</Port>
  </HostOriginEntry>
  <HostOriginEntry>
  <Host>images.example.com</Host>
  <Origin>host2.example.com</Origin>
  <Port>80</Port>
  </HostOriginEntry>
  <HostOriginEntry>
  <Host>video.example.com</Host>
  <Origin>host3.example.com</Origin>
  <Port>80</Port>
  </HostOriginEntry>
</HostOriginMap>
```

Example: Failover Across Origin Servers

- Type of delivery—On-demand (Web content delivery)
- Delivery protocol—HTTP
- Ingest protocol—HTTP
- Origin server details—host1.example.com, host2.example.com, host3.example.com
- Mapping request to origin policies based on domain (example.com) and URI pattern (/web):

- Media Flow Controller sends requests to host1.example.com, by default
- Media Flow Controller fails over to host2.example.com, when host1.example.com returns 404 or 500 error
- Media Flow Controller fails over to host3.example.com, when host1.example.com and host2.example.com return 404 or 500 error
- Media Flow Controller sends constant HTTP probes to the configured " heartbeatpath" in the origin server
- Namespace configuration:

  ```
  namespace example
    domain example.com
    match uri /web
    origin server http server-map OriginServerDefinitions
    delivery protocol http
    status active
  ```

- Server map configuration:

  ```
  server-map OriginServerDefinitions
    format-type origin-escalation-map
    file-url http://example.com/serverdefinitions.xml
    node-monitoring heartbeat
      allowed-fails 3
      connect-timeout 100
      interval 100
      read-timeout 100
  ```

Contents of serverdefinitions.xml:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE OriginEscalationMap SYSTEM "OriginEscalationMap.dtd">
<OriginEscalationMap>
<Header>
<Version>1.0</Version>
<Application>MapXML</Application>
</Header>
  <OriginEscalationMapEntry>
  <Origin>host1.example.com</Origin>
  <Port>80</Port>
  <Options>heartbeatpath=/hello.html,weight=1,
   http_response_failure_codes=404;500</Options>
  </OriginEscalationMapEntry>
  <OriginEscalationMapEntry>
  <Origin> host2.example.com </Origin>
  <Port>80</Port>
  <Options>heartbeatpath=/hello.html,weight=2,
   http_response_failure_codes=404;500</Options>
  </OriginEscalationMapEntry>
  <OriginEscalationMapEntry>
  <Origin>host1.example.com</Origin>
  <Port>80</Port>
  <Options>heartbeatpath=/hello.html,weight=3,
   http_response_failure_codes=404;500;503</Options>
  </OriginEscalationMapEntry>
</OriginEscalationMap>
```

Example: Distribute Load Across Multiple NFS Origins

- Type of delivery—On-demand (Web content delivery)
- Delivery protocol—HTTP
- Ingest protocol—NFS
- Origin server details (NFS path):
  - \\host1.example.com\html
  - \\host2.example.com\images
  - \\host3.example.com\videos
- Mapping request to origin policies based on domain (example.com) and URI pattern (/web).
  - Namespace configuration:

    ```
    namespace example
      domain example.com
      match uri /web
      origin server nfs server-map OriginServerDefinitions
      delivery protocol http
      status active
    ```

  - Server map configuration:

    ```
    server-map OriginServerDefinitions
      format-type nfs-map
      file-url http://example.com/serverdefinitions.xml
    ```

Contents of serverdefinitions.xml:

```
<response scode="0" scode_description="success!">
<PUBLISHINGPOINTS INTERVAL-SEC="3600">
<MISSINGFILE PATH=" "/>
<INVALIDPUBLISHINGPOINT PATH=" "/>
<PUBLISHINGPOINT NAME="name1" PATH="\\host1.example.com\html"/>
<PUBLISHINGPOINT NAME="name2" PATH="\\host2.example.com\images"/>
<PUBLISHINGPOINT NAME="name3" PATH="\\host3.example.com\videos"/>
</PUBLISHINGPOINTS>
</response>
```

Example: Deliver Streams Using Microsoft SmoothStreaming

- Type of delivery—On-demand (video delivery)
- Delivery protocol—HTTP
- Ingest protocol—NFS
- Origin server details—nfs.example.com
- Mapping request to origin policies based on domain (example.com) and URI pattern (/video)

- Sample URL:

  http://www.example.com/videos/bunny.ism/QualityLevels(400000)/
  Fragments(video=134345672)

- Virtual player configuration:

  virtual-player *SmoothStream* type smoothstream-pub
  fragment-tag *Fragments*
  quality-tag *QualityLevels*

- Namespace configuration:

  `namespace example`

  domain *example.com*
  match uri */video*
  origin server nfs *nfs.example.com:/videos/*
  delivery protocol http
  status active
  virtual-player *SmoothStream*

NOTE: You configure Media Flow Publisher with the Web interface.

Example: Deliver Streams Using Adobe Dynamic HTTP Streaming

- Type of delivery—On-demand (video delivery)
- Delivery protocol—HTTP
- Ingest protocol—NFS
- Origin server details—nfs.example.com
- Mapping request to origin policies based on domain (example.com) and URI pattern (/video)

  - Sample URL:

    `http://www.example.com/videos/foo1000Seg1-Frag4`

  - Virtual player configuration:

    `virtual-player FlashStream type flashstream-pub`

    fragment-tag *Frag*
    segment-tag *Seg*
    seg-frag-delimiter –

  - Namespace configuration:

    `namespace example`

    domain *example.com*
    match uri */video*
    origin server nfs *nfs.example.com:/videos/*
    delivery protocol http
    status active
    virtual-player *FlashStream*

Example: Deliver Streams Using Apple HTTP Streaming

No virtual player configuration is required other than the standard namespace configuration settings given in the previous sections.

Related Documentation

- Reverse Proxy Deployment Process on page 450
- Media Flow Controller Deployments Overview on page 447
- Reverse Proxy Deployments on page 448
- Reverse Proxy Cache Tuning CLI Commands on page 452

Transparent Proxy Deployments

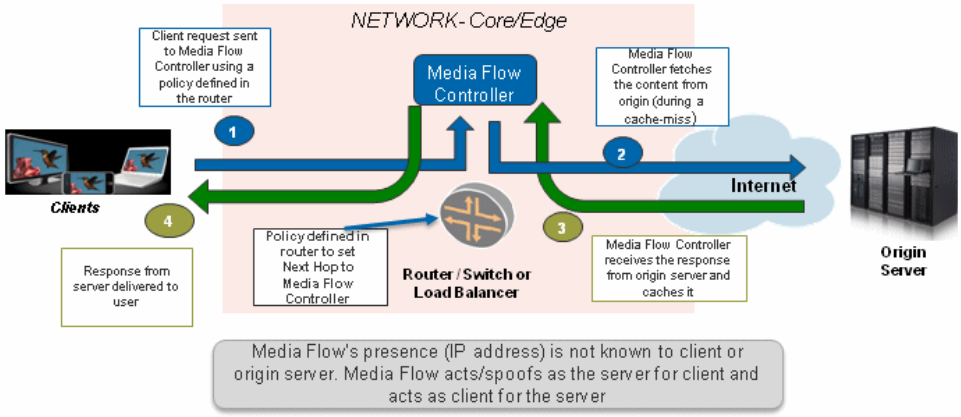
About Transparent Proxies

A transparent proxy, or "Content Direct," solution saves transit bandwidth for network service providers (NSP) and improves subscriber's online experience, through faster downloads. Content Direct saves bandwidth for NSPs, by delivering repeat requests to content from Media Flow Controller. Bandwidth savings achieved using Content Direct varies significantly based on factors such as storage capacity of the Media Flow Controllers, size of objects that get cached (small versus large), type of objects accessed by users (static versus dynamic, cacheable versus non-cacheable), amount of popular content, and number of users accessing content. Deploy Media Flow Controller as a transparent proxy when one or more of the following criteria are met:

- You do not want users or the origin servers to know about the presence of a proxy/cache
- You want to do caching at the edge
- You are an ISP or Campus Edge operator
- You do not want users to modify their browser configuration to point to a proxy
- You do not want to change the DNS configuration
- You are ready for policy-based routing at edge

See Figure 29 on page 463 for a graphic depicting a transparent proxy call flow.

Figure 29: Transparent Proxy Call Flow



Related Documentation

- Transparent Proxy Deployment Requirements on page 463
- Transparent Proxy Deployment Process on page 464
- Transparent Proxy Example Configuration—General IPv6 on page 467
- Transparent Proxy Example Configuration—General IPv6 on page 467
- DMCA Compliance Transparent Proxy Configuration Requirements on page 470

Transparent Proxy Deployment Requirements

You need the following information to get started with the configuration:

- **Type of delivery**—Live or on-demand video delivery, or Web content delivery.
- **Delivery protocol**—HTTP is cached, everything else is tunneled.
- **Ingest Protocol**—HTTP only.
- **Origin server details**—Fully Qualified Domain Name (FQDN), IP address.
- **Mapping request to origin policies**—The list of parameters in the URL (domain, URI, and query parameters) for caching and delivery policies.

To achieve maximum performance, do not use the same device as transparent and reverse proxy. The same Media Flow Controller code base is used for both; however transparent proxy is currently limited by the number of client and origin connections rather than throughput. Media Flow Controller currently can support 100,000 subscriber connections. If the number of required connections for a transparent proxy is close to the limit, use separate reverse and transparent proxy Media Flow Controllers.

A combination of reverse proxy and transparent proxy can be deployed together in the same machine as long as the connection and throughput capacity limits are not exceeded.

Media Flow Controller supports IPv6 for transparent proxy. When the Media Flow Controller's IP address is used to fetch content from the origin, IPv6 clients can fetch

cached content from the Media Flow Controller. If the content requested by an IPv6 client is not cached in the Media Flow Controller, the Media Flow Controller will request the content from the IPv6 origin using the IPv6 IP address obtained from an IPv6 AAAA DNS query.

NOTE: **IPv6 support for transparent proxy is not available when the client IP address is used to fetch content from the origin.**

Related Documentation

- About Transparent Proxies on page 462
- Media Flow Controller Deployments Overview on page 447
- Reverse Proxy Cache Tuning CLI Commands on page 452

Transparent Proxy Deployment Process

We assume that the Media Flow Controller is configured with the basic settings such as license configuration, interface configuration, IP address assignment, hostname, domain list, system clock, DNS, IP routes, and default gateway configuration.

For IPv6 deployments, you must enable IPv6 for the interface configuration. See "Configuring Network Connectivity for IPv6 (CLI)" on page 41 for some basic settings such as interface and default gateway configuration.

Once the basic configurations are done and the required information gathered, you are ready to begin the deployment process, as outlined:

1. Create a namespace—A namespace contains a set of policies that influence how content is fetched from origin server, cached and delivered to users. You can create multiple namespaces if you want to apply separate set of policies for each of the properties, domains, or customers. Examples of transparent proxy configurations are given in "Transparent Proxy Example Configuration—General" on page 466, "Transparent Proxy Example Configuration—General IPv6" on page 467, and "Transparent Proxy Example Configuration—YouTube" on page 468.
2. Map requests to namespaces by configuring a namespace domain and namespace match criteria—See"Creating a Namespace and Setting Namespace Options (CLI)" on page 76 for details on using namespaces.
3. Define origin server settings—For transparent proxy, you configure namespace options to derive the origin IP address from the request from the client. See Table 48 on page 465 for details on the configurations and defaults for transparent proxy deployments.

   For IPv6 support, you configure the **namespace absolute-url** option with the IPv6 address to derive the origin IP address from the client request. Verify that the **namespace ip-version follow-client** option is set to select the IP version of the client for connecting to the origin server.
4. No delivery protocols configuration is needed for IPv4—Transparent proxy supports caching of HTTP only; it tunnels all other protocols.

For IPv6 support, you must enable IPv6 for delivery protocols configuration using the **delivery protocol http ipv6 enable** command.

5. Create virtual-players (if applicable)—The **virtual-player** is the Media Flow Controller video requests' processing engine that performs the following functions:
   - Acts like a media player when fetching content from the origin server.
   - Acts like a media server when delivering content to users.

   Virtual players in a transparent proxy are typically used to perform any of the following functions:
   - Configure query, or video, "seek" related settings.
   - Configure URI processing; see "Configuring the YouTube Virtual Player" on page 129 for details.
   - Define how Media Flow Controller should uniquely identify objects in cache.
6. Tune cache configuration (if required)—Media Flow Controller's default configuration is tuned for optimal caching and delivery performance. You can tune the cache configuration to maximize the performance for the given deployment. Refer to the following sections:
   - "Transparent Proxy Cache Tuning CLI Commands" on page 472—For tuning network connection-related parameters such as maximum bandwidth, connection idle timeout, and concurrent sessions.
   - "Transparent Proxy Cache Tuning Examples" on page 474—For tuning request/response processing-related parameters such as overriding cache options, object size tuning, header insertion/deletion, and enabling connection pooling.

Table 48: Namespace origin-server Settings for Transparent Proxy Type*

origin-server setting	Derive origin from the...	Source IP for Cache Miss	Destination IP for Cache Miss
http follow header HOST	HOST header value	Media Flow Controller IP address	DNS resolved IP address of the origin-server from the HOST header
http follow header *name* use-client-ip†	Specified X-NKN or custom header		DNS resolved IP address of the origin-server from the given header
http follow dest-ip	Client request destination IP	Media Flow Controller IP address	No DNS resolution. Origin IP is client destination IP
http follow dest-ip use-client-ip†	Client request destination IP		No DNS resolution. Origin IP is client destination IP

* All transparent proxy origin-server settings described automatically set origin-request host header inherit incoming-req permit.

† For IPv6, Media Flow Controller does not support the **use-client-ip** option.

Related Documentation

- Configuring Network Connectivity for IPv6 (CLI) on page 41
- Transparent Proxy Example Configuration—General on page 466
- Transparent Proxy Example Configuration—General IPv6 on page 467
- Transparent Proxy Example Configuration—YouTube on page 468
- DMCA Compliance Transparent Proxy Configuration Requirements on page 470
- DMCA Compliance Transparent Proxy Configuration Details on page 470

Transparent Proxy Example Configuration—General

This section describes configurations recommended for transparent proxy deployments.

NOTE: All transparent proxy configurations need to be customized to your specific pattern of traffic. You might need the assistance of Juniper Networks Support to fine-tune your cache performance.

From Release 11.A.1 onwards, Media Flow Controller changed the namespace configuration requirements for transparent proxy deployments. The **namespace** *name* **proxy-mode** option is deprecated and no longer available. Two new options, **delivery protocol http transparent** *client_traffic_NIC* **enable** and **namespace** *name* **origin-server http follow** *option* **use-client-ip**, are used instead.

1. Set and enable delivery interfaces for transparent proxy:
   a. Set HTTP delivery interfaces:

      delivery protocol http interface *client_traffic_NIC*

   b. Enable those interfaces for transparent proxy:

      delivery protocol http transparent *client_traffic_NIC* **enable**

2. Create a namespace that inherits the host header from the client request when making the origin request for transparent proxy, disable setting the " X-Forwarded-For" header to the value of the client IP address when requests are sent from Media Flow Controller to origin upon a cache miss, and set **match**, **precedence**, and **origin-server** options.

   ```
   namespace tproxy
     delivery protocol http origin-request host-header inherit incoming-req permit
     delivery protocol http origin-request x-forwarded-for disable
     match uri / precedence 10
     origin-server http follow header host use-client-ip
   ```

3. Set an **object-size** limit on storing media objects for namespace **tproxy**; objects below this size go into memory, and never go to any disk or SSD. Also, set the **cache-fill** option so only requested data is fetched. Finally, set **302-revalidate** to handle cookie revalidation requests. From **namespace** prefix mode:

   ```
   delivery protocol http origin-fetch content-store media object-size 4096
   delivery protocol http origin-fetch cache-fill client-driven
   ```

```
delivery protocol http origin-fetch redirect-302 handle
```

4. Optionally, set a lower sync interval for hotness data in the cache; default is **14400** seconds. This value should be decreased for transparent proxy caching because the number of objects and the rate of change of those objects is much higher than in the reverse proxy mode.

   ```
   ram-cache sync interval integer
   ```

5. Optionally, define the maximum allowed number of TIME_WAIT sockets held by the system simultaneously; this can be useful when running non-persistent connections or performance tests on transparent proxy configurations.

   ```
   network tcp max-tw-buckets number
   ```

Related Documentation

- Transparent Proxy Deployment Process on page 464
- Transparent Proxy Example Configuration—General IPv6 on page 467
- Transparent Proxy Example Configuration—YouTube on page 468
- DMCA Compliance Transparent Proxy Configuration Requirements on page 470
- DMCA Compliance Transparent Proxy Configuration Details on page 470

Transparent Proxy Example Configuration—General IPv6

This section describes configurations recommended for transparent proxy deployments supporting IPv6.

NOTE: All transparent proxy configurations need to be customized to your specific pattern of traffic. You might need the assistance of Juniper Networks Support to fine-tune your cache performance.

1. Set and enable delivery interfaces for transparent proxy.

   a. Set HTTP delivery interfaces:

      ```
      delivery protocol http interface client_traffic_NIC
      ```

   b. Enable those interfaces for transparent proxy:

      ```
      delivery protocol http transparent client_traffic_NIC enable
      ```

   c. Enable those interfaces for IPv6:

      ```
      delivery protocol http ipv6 enable
      ```

2. Create a namespace that always uses IPv6 to connect to the origin server, and set the **match** option for virtual host.

   ```
   namespace tproxy
     origin-server http ip-version follow-client
     match virtual-host ip_address
   ```

Related Documentation

- Transparent Proxy Deployment Process on page 464

Transparent Proxy Example Configuration—YouTube

This section describes configurations found to be beneficial for transparent proxy deployments for YouTube media. You can find additional configurations and information at "Configuring the YouTube Virtual Player" on page 129.

1. If you install with root cache; disable it (disk **dc_1**) because this is the root disk and Media Flow Controller does a lot of logging on that disk—it shouldn't be used for caching.

   media-cache disk dc_1 status inactive

2. Disable cache promotion for the following reasons:

   a. Hotness promotion algorithm keeps track of only 200 K objects. At 1 Gbps, we typically see 2.4 M objects per day, so 200 K is not enough.

   b. Promotion does not take size into account, so a large object could be promoted and cause a large number of small objects to be evicted from SSD.

      analytics cache-promotion disable

3. Set a cache ingest size threshold (for example, 2 MB); this, combined with a namespace configuration of **object-size 4096** (given in Step 11), means that objects with size greater than 4K and less than or equal to 2 MB go to SSD and objects with size with greater than 2 MB go to SAS.

   namespace name1 delivery protocol http origin-fetch content-store media cache-tier highest object-size *<value>*

4. Ensure that the RAM cache size setting is zero to allocate 22 to 23 Gigabyte to RAM in a system with 36 Gigabyte RAM. The default is **0** (zero).

   ram-cache cache-size-MB 0

5. Set the queue sizes per thread for each tier. There are four threads per disk. If more than the following number of requests are queued in a disk tier, the object is re-fetched. Default is set at **0** (zero), which means an infinite number of requests can be queued.

   media-cache disk cache-tier sas admission threshold 20
   media-cache disk cache-tier sata admission threshold 12
   media-cache disk cache-tier ssd admission threshold 1250

6. Set and enable delivery interfaces for transparent proxy:

   a. Set HTTP delivery interfaces:

      delivery protocol http interface *client_traffic_NIC*

   b. Enable those interfaces for transparent proxy:

```
delivery protocol http transparent client_traffic_NIC enable
```

7. To tunnel YouTube seeks, use this **virtual-player** configuration:

```
virtual-player youtube_player type youtube
  cache-name video-id query-string-parm "id" format-tag query-string-parm "itag"
  seek-config query-string-parm begin tunnel-mode enable
  exit
```

8. To cache YouTube traffic, use this **namespace** configuration; first, create the namespace with **domain**, **match**, and **origin-server** values; and your **virtual-player**:

```
namespace youtube
  domain regex "^.*\.c\.youtube\.com|^.*\.googlevideo\.com"
  match uri /videoplayback precedence 5
  origin-server http follow header host use-client-ip
  virtual-player youtube_player
```

9. Set the namespace to override cache-control headers, inherit the host header from the client request when making the origin request for transparent proxy, and disable setting the " X-Forwarded-For" header to the value of the client IP address when requests are sent from Media Flow Controller to origin upon a cache miss. From **namespace** prefix mode:

```
delivery protocol http origin-fetch content-store media cache-agethreshold 300
delivery protocol http origin-request host-header inherit incoming-req permit
delivery protocol http origin-request x-forwarded-for disable
```

10. Because YouTube content comes with a " Cache-Control: Private" header, you must set namespace **youtube** to disregard all cache-control headers. From **namespace** prefix mode:

```
delivery protocol http origin-fetch cache-directive no-cache override
```

11. Set an **object-size** limit on storing media objects for namespace **youtube**; objects below this size go into memory, and never to any disk or SSD. This setting may not matter for YouTube since all objects are typically greater in size. From **namespace** prefix mode:

```
delivery protocol http origin-fetch content-store media object-size 4096
```

12. Allow namespace **youtube** to stop downloading an object after the client stops viewing it. If an object is partially cached, then on a second subscriber request, the remainder object is downloaded using a byte range request. If the origin doesn't support byte-range requests, it sends the whole object and Media Flow Controller discards the part that has already been stored. Once done, exit namespace **youtube** and save. From **namespace** prefix mode:

```
delivery protocol http origin-fetch cache-fill client-driven
```

13. Add the **virtual-player** you configured; activate the namespace, exit, and save your configuration. From **namespace** prefix mode:

```
virtual-player youtube_player
status active
exit
write memory
```

Related Documentation

- Configuring the YouTube Virtual Player on page 129
- Transparent Proxy Deployment Process on page 464
- Transparent Proxy Example Configuration—General on page 466
- Transparent Proxy Example Configuration—YouTube on page 468
- DMCA Compliance Transparent Proxy Configuration Details on page 470
- DMCA Compliance Transparent Proxy Configuration Requirements on page 470

DMCA Compliance Transparent Proxy Configuration Requirements

Table 49 on page 470 shows Media Flow Controller compliance with the Digital Millennium Copyright Act (DMCA).

Table 49: DMCA Requirements to Media Flow Controller Functionality

DMCA Service Provider Caching Requirements	Media Flow Compliance
Content cached must not be modified by the service provider.	Content is delivered as-is—not modified.
Provider must comply with rules of "refreshing."	HTTP 1.1 compliant freshness, also known as expiry, management.
Transactions must not interfere with technology that tracks "hits."	Media Flow Controller maintains complete transparency between client and origin and does not interfere with industry-standard technology to track " hit" information.
Limit user's access based on conditions set by the origin server.	HTTP 1.1 compliance ensures that content marked as "Private" is not cached.
Service provider must remove cached content when notified.	XML-API and CLI provide for content deletion and content invalidation.

Related Documentation

- Transparent Proxy Example Configuration—General IPv6 on page 467
- Transparent Proxy Deployment Process on page 464
- Transparent Proxy Example Configuration—General on page 466
- Transparent Proxy Example Configuration—YouTube on page 468
- DMCA Compliance Transparent Proxy Configuration Details on page 470

DMCA Compliance Transparent Proxy Configuration Details

Configuration details for DMCA compliance are:

1. Create a **namespace** that inherits the host header from the client request when making the origin request for transparent proxy. See Table 48 on page 465 for examples.

   origin-request host-header inherit incoming-req permit

This namespace setting, required for all transparent proxy configurations, tells Media Flow Controller to use the incoming header in the request to the origin server; this achieves full transparency to the client and the origin.

2. Disable setting the “X-Forwarded-For” header to the value of the client IP address when requests are sent from Media Flow Controller to origin upon a cache miss. This is done for full transparency.

   ```
   origin-request x-forwarded-for disable
   ```

3. By default, all “Cache-Control” headers are followed by Media Flow Controller. For example, if a “Cache-Control” header says “No-Cache” , Media Flow Controller will not cache the content. No configuration is required.
4. By default, Media Flow Controller does not cache content that has query strings. Query strings are typically used for dynamically changing objects; therefore, Media Flow Controller does not cache query strings. No configuration is required.
5. Revalidate-always: For Media Flow Controller to make sure that the object is always ‘fresh’, it needs to be revalidated every time there is a HTTP request for that object.

   ```
   client-request cache-hit action revalidate-always
   ```

6. By default, Media Flow Controller tunnels authenticated content. No configuration is required.

Related Documentation

- Transparent Proxy Example Configuration—General IPv6 on page 467
- Transparent Proxy Deployment Process on page 464
- Transparent Proxy Example Configuration—General on page 466
- Transparent Proxy Example Configuration—YouTube on page 468
- DMCA Compliance Transparent Proxy Configuration Requirements on page 470

Example: DMCA Compliance Transparent Proxy Configuration

CLI configuration:

```
namespace tproxy
  delivery protocol http origin-request host-header inherit incoming-req
  permit
  delivery protocol http origin-request x-forwarded-for disable
  delivery protocol http client-request cache-hit action revalidate-always
  match uri / precedence 10
  origin-server http follow header host use-client-ip
```

Related Documentation

- Transparent Proxy Example Configuration—General IPv6 on page 467
- Transparent Proxy Deployment Process on page 464
- Transparent Proxy Example Configuration—General on page 466
- Transparent Proxy Example Configuration—YouTube on page 468
- DMCA Compliance Transparent Proxy Configuration Details on page 470

Transparent Proxy Cache Tuning CLI Commands

See Table 50 on page 472 for guidelines on cache performance tuning for transparent proxy deployments.

Table 50: Cache Performance Tuning Settings for Transparent Proxy

Config Parameter	Description	Benefit	Guidelines
analytics cache- ingest size-threshold 65536	Set the maximum size of an object that can be optionally ingested into the fastest cache tier in the disk cache. Objects smaller than, or equal to, the configured size are automatically written to the fastest cache tier. The default, **0**, disallows any object being ingested directly to the fastest tier.	Allow objects to be ingested directly to the fastest cache tier.	This configuration means that: • Objects with size with >64 K go to SAS • Objects with size with >4 K and <=64 K go to SSD • Objects with size with <4 K go to RAM SSD has better performance for smaller objects than SAS, which handles larger objects better.
analytics cache-promotion disable.	Alternatively, in an environment where there is constant churn of objects in the cache, disable cache promotion entirely. Constant churn of cached objects may occur due to "short-lived" objects in the cache (due to shorter object expiry time, or highly diversified requests).	More accurate eviction processing so that cold objects do not waste disk space.	The hotness promotion algorithm keeps track of only 200 K objects. At 1 Gbps, we see 2.4 M objects per day, so 200 K is not enough. Promotion does not take size into account, so a large object could be promoted and cause a large number of small objects to be evicted from SSD.
origin-fetch cache-ingest hotness-threshold	Set a threshold for object "hotness." When the hotness value of an object reaches the threshold value, the object is ingested to the slowest tier. The default value is 3, which is the lowest possible hotness value for an object to get ingested after the first hit.	Delaying objects from moving to slower cache tiers ensures that "hot" objects are served from the fastest cache tier, improving cache efficiency, and bandwidth savings.	By default, when an object is requested for the first time, Media Flow Controller sets the hotness of the object to 3 and the object is ingested to the slowest cache tier.
media-cache disk *dc_1* status inactive	Disable the root disk (for example, **dc_1**) and Media Flow Controller does a lot of logging on that disk—we recommend not using it for caching.	A transparent proxy access log utilizes more disk I/O and should have its own disk.	Recommended for transparent proxy deployments.

Table 50: Cache Performance Tuning Settings for Transparent Proxy *(continued)*

Config Parameter	Description	Benefit	Guidelines
media-cache disk cache-tier sas admission threshold 20 **media-cache disk cache-tier sata admission threshold 12** **media-cache disk cache-tier ssd admission threshold 1250**	Set the queue sizes per thread for each tier. Default is set at **0** (zero), which means an infinite number of requests can be queued.	Better latency when large numbers of requests hit the disk.	There are four threads per disk. If more than the allowed number of requests are queued in a disk tier, the object is fetched from the origin server. However, setting it to these values has been shown to provide better disk utilization.
namespace *name* delivery protocol http origin-fetch cache-directive no- cache override	Override the "Cache-Control" header. The "Cache-Control" header influences the caching behavior of Media Flow Controller. By default, Media Flow Controller does not cache objects that are marked as "No-Cache" or " Private" .	Achieve better bandwidth savings by caching YouTube videos.	This configuration is a violation of HTTP 1.1 standard and should be applied only for select Web portals whose media delivery methods are well understood by the customer. Required for caching YouTube videos.
namespace *name* delivery protocol http origin-fetch cache-fill client- driven	Allow the namespace to stop downloading an object after the client stops viewing it. If an object is partially cached, then, on a second subscriber request, the remainder object is downloaded using a byte range request. If the origin doesn't support byte-range requests, it sends the whole object and Media Flow Controller discards the part that has already been stored.	Optimize the number of bytes that are requested from the origin server, resulting in better bandwidth savings.	Recommended for transparent proxy deployments.
namespace *name* delivery protocol http origin-fetch content-store media cache-age-threshold 300	Increase the amount of time objects can be stored in RAM cache. Default: Objects of age under **60** seconds are stored in RAM.)	Save disk I/O bandwidth by storing objects in RAM for a longer period before moving to disk.	Recommended for transparent proxy deployments.
namespace *name* delivery protocol http origin-fetch content-store media object-size 4096	Also, set an **object-size** limit on storing media objects for namespace **tproxy**; objects below this size go into memory, and never go to any disk or SSD.	Improve disk-cache performance since small objects need not be written into disk and can be served directly from the RAM cache.	Increase the value of **object-size**, to force only large objects to be served from disk.
namespace *name* delivery protocol http origin-request host-header inherit incoming-req permit	Set the namespace to use the " Host" header from the client request when making the origin request for transparent proxy.	The subscriber's client (the browser or another application) can specify the header to be used for requests to the origin.	Required for transparent proxy deployments.

Table 50: Cache Performance Tuning Settings for Transparent Proxy *(continued)*

Config Parameter	Description	Benefit	Guidelines
namespace *name* **delivery protocol http origin-request x-forwarded-for disable**	Set the namespace to disable setting the "X-Forwarded-For" header to the value of the client IP address when requests are sent from Media Flow Controller to origin upon a cache miss.	To be completely transparent, the cache should forward requests as-is from the subscriber's client.	Required for transparent proxy deployments.
ram-cache cache- size-MB 0	Ensure that the RAM cache size setting is **0** (zero), the default, to allocate 22 to 23 Gigabyte to RAM in a system with 36 Gigabyte RAM.	Increases the number of TCP connections that Media Flow Controller can support.	Do not change this configuration from the default: **0** (zero).

Related Documentation

- Transparent Proxy Cache Tuning Examples on page 474
- Transparent Proxy Example Configuration—General IPv6 on page 467
- Transparent Proxy Deployment Process on page 464
- Transparent Proxy Example Configuration—General on page 466
- Transparent Proxy Example Configuration—YouTube on page 468
- DMCA Compliance Transparent Proxy Configuration Details on page 470

Transparent Proxy Cache Tuning Examples

This section provides some guidelines for fine tuning Media Flow Controller's configuration to achieve better bandwidth savings. You can improve bandwidth savings by tuning the following configurations in Media Flow Controller:

- Example: Virtual Player Tuning on page 474
- Example: Object Size Tuning on page 475
- Example: Mime-Type Tuning on page 475
- Example: Cache Age Tuning on page 476
- Example: Popularity Tuning on page 476
- Example: Cache-Control Override on page 476

Example: Virtual Player Tuning

In today's Internet, Web portals represent a single media asset using multiple URLs. A unique URL is generated for every access. Web portals do that for reasons such as prevention of content thefts, securing servers from DOS attacks, and prevention of browser/player caching. However, this poses a number of challenges to NSP edge caching solutions. A media asset may be identified by a query parameter in the URL, instead of the entire URL. Given below is an example of how YouTube generates URLs. The highlighted query parameters represent the uniqueness of a video delivered from YouTube.

```
GET "http://v8.nonxt7.c.YouTube.com/
videoplayback?ip=0.0.0.0&sparams=id%2Cexpire%2Cip%2Cipbits%2Citag%2Calgorith
```

```
m%2Cburst%2Cfactor&fexp=904405&algorithm=throttle
factor&itag=34&ipbits=0&burst=40&sver=3&expire=1266310800&key=yt1&signatu
re=66222E9350B9BB5AC68297F12AC1DCB4C53AAFDE.55B33FFFA04EBF001AF39A4F316E657F
C318E0E5&factor=1.25&id=efa3a0434887fdc0&redirect_counter=1”
```

Media Flow Controller uses the **virtual-player** configuration to uniquely identify objects, though the objects may be represented using different URLs, because **virtual-player** allows you to configure parameters in the URL to represent the object identity. For example, to enable YouTube caching, the following configuration (variables italicized) needs to be applied to the Media Flow Controller:

```
virtual-player ytplayerA type youtube
  cache-name video-id querystring-parm id format-id query-string-parm itag
  seek query-string-parm begin
```

You can define virtual players for popular websites such as YouTube, Rapidshare, Dailymotion, and Youko, so Media Flow Controller can cache objects delivered from those Web portals. For a list of complete websites that can be cached by Media Flow Controller and its corresponding **virtual-player** configuration, please contact cmbu-support@juniper.net.

Example: Object Size Tuning

Media Flow Controller stores media objects in a cache file-system that includes RAM-cache as well as disk-cache (SSD, SAS, and SATA drives). There may be deployments where the traffic pattern consists of a mix of small objects (such as Web pages, graphics, icons) and large objects (such as videos, software installation packages).

Storing small objects in disk-cache has a performance overhead, because disk read/write operations are expensive (when compared to RAM read/write operations). You can tune the cache configuration so that small objects are just cached in RAM and delivered to users. Small objects need not be written to disk, thus freeing up the disk-cache for large objects and also reducing the latency for small objects delivery.

Use the namespace **origin-fetch content-store media object-size** CLI command to tune the size of objects that get served from disk. For example, **origin-fetch content-store media object-size 4096** would force Media Flow Controller to store in disk-cache all fetched objects larger than 4 KB.

Use the **cache-tier highest object-size** CLI command to set the maximum size of an object that can be optionally ingested into the fastest cache tier, such as SSDs. Objects smaller than, or equal to, the configured size are automatically promoted to the fastest cache tier.

Example: Mime-Type Tuning

Media Flow Controller, by default, can cache all static media objects. However, based on the traffic profile of the service provider network, Media Flow Controller can be configured to “selectively” cache objects. For example, Media Flow Controller can be configured to cache only “video” objects and “software installation packages.” To do this, you define namespaces so that Media Flow Controller caches only video files (such as *.mpg, *.wmv, *.mov, and *.flv) and software installation packages (such as *.iso, *.vdf, *.msu, and *.exe). You define the files to be cached by Media Flow Controller using the **namespace match uri** CLI command.

Example: Cache Age Tuning

Media Flow Controller, by default, caches objects of expiry time 60 seconds or less, in RAM. The expectation is that these objects will be evicted soon and they are not worth storing in disk caches (wasting the disk I/O cycles). In transparent proxy deployments, it is recommended to increase this value to 300 seconds or higher, in order to improve the disk cache efficiency. Customers can use the namespace **origin-fetch content-store media cache-age-threshold 300** configuration to tune the cache age.

Example: Popularity Tuning

The Media Flow Controller cache-management algorithm automatically promotes popular or "hot" objects (the most requested objects) to a faster cache tier. "Cold" objects (the least requested objects) remain in slower cache tiers.

You can influence when an object gets promoted to a faster cache tier. Delaying objects from moving to slower cache tiers ensures that "hot" objects are served from the fastest cache tier, improving cache efficiency, which indirectly improves bandwidth savings. By default, when an object is requested for the first time, it becomes a candidate for promotion to the slowest cache tier. Use the **origin-fetch cache-ingest hotness-threshold** CLI command to set a threshold for "hotness" value after which an object is ingested to the slowest cache tier.

In an environment where there is constant churn of objects in the cache, we recommend disabling cache promotion. Constant churn of cached objects may occur due to " short-lived" objects in the cache (due to shorter object expiry time, or highly diversified requests). Disabling cache promotion prevents objects getting promoted to a faster cache tier—especially in circumstances where they get evicted immediately after the promotion. Disable cache promotion with the CLI command **analytics cache-promotion disable**.

Example: Cache-Control Override

The "Cache-Control" header influences the caching behavior of Media Flow Controller. By default, Media Flow Controller does not cache objects that are marked as "No-Cache" or "Private" . However, service providers may wish to selectively override the Cache-Control header to achieve better bandwidth savings. Use the **cache-directive no-cache override** CLI command to override the Cache-Control header. This configuration is a violation of HTTP 1.1 standard and should be applied only for select Web portals whose media delivery methods are well understood by the customer.

Related Documentation

- Transparent Proxy Cache Tuning CLI Commands on page 472
- Transparent Proxy Example Configuration—General IPv6 on page 467
- Transparent Proxy Deployment Process on page 464
- Transparent Proxy Example Configuration—General on page 466
- Transparent Proxy Example Configuration—YouTube on page 468
- DMCA Compliance Transparent Proxy Configuration Details on page 470

Mid-Tier Proxy Deployments

Service providers are using three-tier architectures de-coupling front end servers from back-end storage servers by deploying mid-tier proxies. Mid-tier proxies reduce network latency, save bandwidth costs, offload origin servers, and scale front-end server throughput. Media requests are handled by front-end servers, supporting multiple protocols, that issue HTTP fetch requests to Media Flow Controller mid-tier proxy, instead of to origin servers. Media Flow Controller serves the content just as an origin server would in a network without mid-tier servers.

You can configure Media Flow Controller as a **mid-tier** proxy using a **namespace** setting. In this deployment, you must also configure user browsers to point at the Media Flow Controller; **domain** and **match uri-prefix** settings are generally irrelevant and can be set to **any** and **/** (slash), respectively. If you have multiple namespaces, set a low precedence (**10** is the lowest) so incoming requests can pass to namespaces with specific **uri-prefix** matches (instead of **/** that accepts all incoming requests).

1. Create a namespace with **domain any**, **match uri / precedence 10** and **origin-server http absolute-url**.
2. Configure your browser to point at the Media Flow Controller. In Internet Explorer, you do this through the **Tools > Internet Options > Connections > LAN settings** page. In Mozilla Firefox, use the **Tools > Advanced > Network > Connection Settings** page.

In this way, any connections to the Internet go through Media Flow Controller to the appropriate namespace rather than this one.

Related Documentation

- Media Flow Controller Deployments Overview on page 447
- About Transparent Proxies on page 462
- Transparent Proxy Cache Tuning CLI Commands on page 472
- DMCA Compliance Transparent Proxy Configuration Requirements on page 470
- Transparent Proxy Deployment Process on page 464

PART 2

Index

Index

D

N

O

P

R

S

V

W