Product Overview

This chapter introduces the Cisco Videoscape Distribution Suite, Internet Streamer (VDS-IS).

Overview

The Cisco VDS-IS is a distributed network of Content Delivery Engines (CDEs) running Content Delivery Applications (CDAs) that collaborate with each other to deliver multi-format content to a variety of client devices. The client devices supported are personal computers and Wi-Fi-enabled mobile devices, such as personal digital assistants (PDAs).

The VDS-IS supports a variety of mechanisms to accelerate the distribution of content within the content delivery network. It also offers an end-to-end solution for service providers to ingest and stream entertainment-grade content to subscribers.

The VDS-IS functionality can be separated into four areas:

- Ingest
- Distribution
- Delivery
- Management

Each CDE in the VDS-IS contributes to one or more of these functions as determined by the CDAs running on it. Table 1-1 describes the relationship between the CDA names and the Internet Streaming Content Delivery System Manager (CDSM) device names.

Table 1-1 CDA Mapping to Functionality and CDSM

CDA Name	Functionality	CDSM Device Name
Internet Streamer (+ Content Acquirer)	Ingest, distribution, and delivery	Service Engine (SE)
Service Router	Redirect client requests for delivery	Service Router (SR)
Internet Streaming Content Delivery System Manager	Management	CDSM

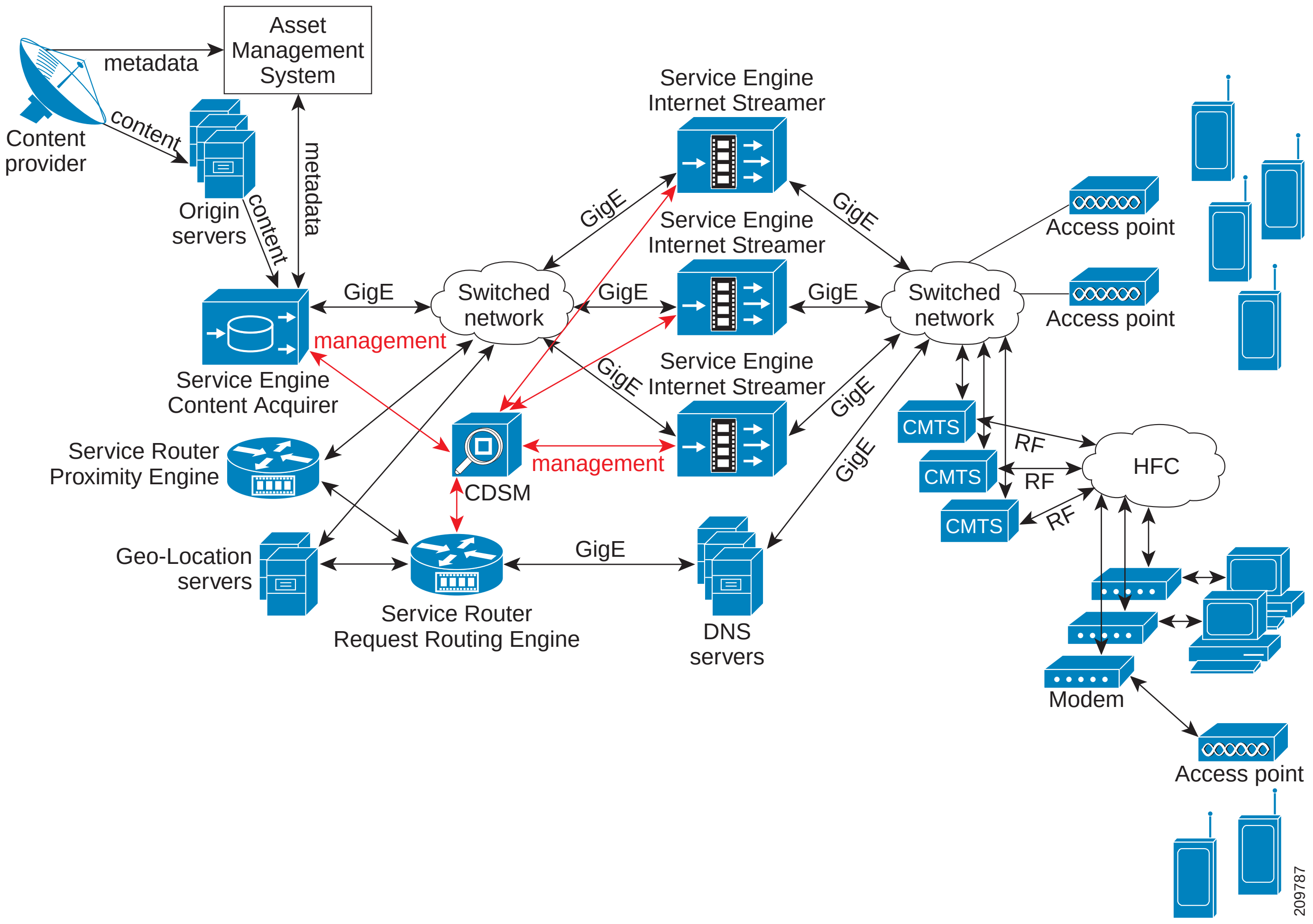
The Service Engine can function as a Content Acquirer and Internet Streamer, or just as an Internet Streamer.

Figure 1-1 shows the major elements of a VDS-IS network. How content flows, from ingest to distribution within the VDS-IS, to delivery to client devices, is dictated by the content delivery services defined for each content origin. A Delivery Service is a configuration defined by using the CDSM and consists of configuration parameters that dictate how content is ingested and distributed, and what content is delivered to the client devices. Some of the primary Delivery Service definition parameters are as follows:

- Origin server
- Service routing domain name
- Service Engines participating in the Delivery Service
- Service Engine designated as the Content Acquirer

The Content Acquirer is only active on one Service Engine in each Delivery Service.

Figure 1-1 High-Level View of the Cisco VDS-IS



The following sections briefly describe the elements of the VDS-IS. For more detailed information, see the "Content Delivery System Architecture" section on page 1-5.

Ingest and Distribution

The Service Engine designated as the Content Acquirer for a Delivery Service is the ingest device. VDS-IS supports the following methods of content ingest:

- Prefetch ingest
- Dynamic ingest
- Hybrid ingest
- Live stream ingest and split

The distribution of content within the VDS-IS is determined by the method of ingest used.

Prefetch Ingest

The Content Acquirer receives metadata from the back-office in the form of an XML-formatted Manifest file, and using the information in the file, pulls the content into storage on the Content Acquirer. The content can be ingested by using different protocols. The supported protocols are FTP, HTTP, HTTPS, and CIFS, which are files copied to the Service Engine. The ingested content is then distributed to all Service Engines in the content Delivery Service. The content is stored on each Service Engine's hard disk for a configurable amount of time or until the content entry gets deleted from the Manifest file. This is called *content pinning*.

The Manifest file can be used to specify different policies for content ingest and also for streaming the prefetched content. For example, the policy could include specifying the expiry of the content, setting time windows in which the content is made available to users, and so on.

Note The content type (MIME) value cannot exceed 32 characters.

Dynamic Ingest

Content can be dynamically ingested into the VDS-IS. Dynamic ingest is triggered when a Service Engine's Internet Streamer application does not find a client's requested content in its local hard disk storage. All Service Engines participating in the content Delivery Service coordinate to form a content distribution tunnel starting at the origin server and ending at the Service Engine responding to the client request. As the content flows through this tunnel, the participating Service Engines cache a copy of the content. Subsequent requests for the same content are served off the VDS-IS network. Content ingested and distributed by this method is deleted if clients do not request it frequently.

The Internet Streaming CDSM manages this ingest method internally, not by instructions embedded in a Manifest file, and manages the storage automatically. The Internet Streaming CDSM also provides the ability to purge any dynamically ingested content out of the Service Engines. Content is identified by a URL, which is also used to delete the content.

Hybrid Ingest

The hybrid ingest method provides a very powerful solution by combining the features of the prefetch ingest and the dynamic ingest methods. The metadata and control information about the content, defined in the Manifest file, is propagated and pinned to all Service Engines participating in the content Delivery Service. However, the content is not prefetched. Ingest occurs upon user request for the content. Content that is cached on the Service Engines by using this method is subject to the same deletion rules as the dynamic ingest method. The metadata that is propagated can be used to specify explicit controls and policies for streaming the content.

Live Stream Ingest and Split

The live stream ingest method distributes a live content feed to all of the Service Engines participating in the content Delivery Service and helps to scale the content delivery to a very large audience. This method leverages the live stream splitting capabilities of the Internet Streamer application and optimizes the access by doing a one-to-many split to all Service Engines in the content Delivery Service. The Internet Streaming CDSM provides the necessary interface to schedule the streaming of live programs. Advanced techniques are used to enhance the performance of live streaming.

Delivery

The Service Router handles client requests for content and determines the best Service Engine to deliver it based on proximity, load and health states.

Once the best Service Engine has been determined, the content is delivered to the client device by means of one of the following mechanisms:

- **Static Content Download using HTTP**—Content is downloaded by the client device before it can be rendered to the user.
- **Progressive Content Download using HTTP**—Content is rendered in segments to the user before it has been fully downloaded.
- **Content Streaming using HTTP, RTMP, RTSP, or RTP**—Content is streamed to the client device, Service Engines collect feedback and can fine-tune streaming. Advanced error recovery can also be performed. This is a very common method of streaming video content to client devices.

Table 1-2 lists the content types and formats, content transport protocols, and client types supported by the VDS-IS.

Table 1-2 Supported Content Types

Content Types and Formats	Transport Protocols	Typical Client Types	Access Network Type
Windows Media (WMA, WMV, ASF, and others) VC-1	RTP, RTSP, HTTP	Windows Media Player 9, 10, 11 on PC Windows Media Player 9 for Mac Windows Media Technology (WMT) Silverlight	Wired Wi-Fi Cellular
QuickTime (MOV), hinted (3GP) files	RTP, RTSP, HTTP	On PC: QuickTime Player, QuickTime Pro 6 or 7, RealPlayer 10 or 11 (3GP only), VLC player On Mac: QuickTime Player, QuickTime Pro 6 or 7, RealPlayer 10 for Mac OS X (3GP only)	Wired Wi-Fi Cellular

Table 1-2 Supported Content Types (continued)

Content Types and Formats	Transport Protocols	Typical Client Types	Access Network Type
Other Hypertext and image files (HTML, JPEG, and so on)	HTTP	Web browsers and other HTTP clients	Wired Wi-Fi Cellular
MPEG (MP1, MP2, MP4)	RTP, RTSP	MPEG clients **Note** For Flash Media Streaming, the Adobe Flash Media Player 9 update 3, Adobe Media Player, and Adobe Air, are the only players that support MPEG-4.	Wired
Adobe Flash (SWF, FLV, MP3)	RTMP, HTTP	Adobe Flash Player 9 for Windows, Mac OS, and Linux	Wired Wi-Fi Cellular
H.264	RTMP, HTTP	H.264 clients **Note** For Flash Media Streaming, the Adobe Flash Media Player 9 update 3 is the only supported player.	Wired

Note RTMP is part of the Flash Media Streaming feature.

Management

The Internet Streaming CDSM, a secure web browser-based user interface, is a centralized system management device that allows an administrator to manage and monitor the entire VDS-IS network. All devices, Service Engines and Service Routers, in the VDS-IS are registered to the Internet Streaming CDSM.

Service Engines can be organized into user-defined device groups to allow administrators to apply configuration changes and perform other group operations on multiple devices simultaneously. One device may belong to multiple device groups.

The Internet Streaming CDSM also provides an automated workflow to apply a software image upgrade to a device group.

Content Delivery System Architecture

The VDS-IS consists of an Internet Streaming CDSM, one or more Service Engines, and one Service Router. For full redundancy, a VDS-IS would include an additional CDSM and Service Router. The Service Engine handles content ingest, content distribution within the VDS-IS, and content delivery to client devices. The Service Router handles client requests and redirects the client to the most appropriate Service Engine. The Internet Streaming CDSM manages and monitors the VDS-IS, the delivery services, and all of the devices in the VDS-IS.

- Service Engine
- Service Router
- Content Delivery System Manager
- Resiliency and Redundancy

Service Engine

Each Service Engine can function as a Content Acquirer and Internet Streamer, or just as an Internet Streamer. Based on the Service Engines' assignments to different delivery services, the right set of applications supporting the functions is enabled. For example, only one Service Engine is assigned the role of Content Acquirer in each Delivery Service. In addition, the Service Engine assigned as the Content Acquirer in a Delivery Service also includes the functions of an Internet Streamer.

Storage and Distribution

Both the Content Acquirer and the Internet Streamer applications have storage and distribution functions within the VDS-IS, which include the following:

- Management of the physical storage of content and metadata. Content URLs are translated into their physical file paths for content retrieval, deletion, and update.
- Management of dynamically ingested content and periodic replacement of content not accessed frequently. Content replacement is performed by sophisticated content-replacement algorithms. The algorithms add *weight* to the content according to size, frequency of access, and other attributes to produce the list of content that needs to be purged.
- Ingest of prefetched content and retrieval of such content for distribution to other Service Engines in the same Delivery Service.
- Maintenance of information about the entire VDS-IS topology and all of the delivery services. This includes upkeep of a list of Service Engines in the same Delivery Service that is used for distributing prefetched, dynamic, and live stream content.
- Maintenance of the database that stores and distributes metadata about the content, and the topology and Delivery Service information.
- Distribution of content on a per-Delivery Service basis, where the flow path of content could differ from one Delivery Service to another.

FastCAL

The Content Abstraction Layer (CAL) library provides an interface to the Content Delivery Network File System (CDNFS). The CAL library monitors the content in the CDNFS and communicates with the Content Manager process to evict less popular content.

The Fast Content Abstraction Layer (FastCAL) library provides quick response time for high-performance Web Engine create, update, lookup, and delete operations. All other protocol engines and modules, including live streaming for Flash Media Streaming and RTSP gateway, continue to use the CAL library and Unified Namespace (UNS) process. Flash Media Streaming VOD (prefetched, hybrid and dynamically cached content) use FCAL by way of the Web Engine. FastCAL communicates with the Content Manager for popularity tracking. Lookup notifications are also sent from FastCAL to the Content Manager.

Disk Path

FastCAL creates the disk path for cache content. An example of a disk path with an HTTP content URL of http://192.168.1.9/vod/foo.flv follows:

/disk11-01/c/192.168.1.9/1d/a1/1da1394af838bbcb45af78fd5681abeb/foo.flv.http

The disk path for the prefetched HTTP content URL, http://192.168.1.9/vod/c5.flv, translates to the following:

/disk03-01/p/192.168.1.9/1d/a1/1da1394af838bbcb45af78fd5681abeb/c5.flv

Disk Allocation

Disk usage for all disks is maintained in shared memory, which is updated by the Content Manager with actual disk usage and by FastCAL when new content is created. FastCAL creates predefined *buckets*, which are groups of disks. The number of disks per bucket varies, and the number of buckets varies; it is determined by the CDE model.

The CDNFS disk mount point is always displayed as disk***XX-YY***, where ***XX*** is the disk number and ***YY*** is the partition. Every content URL is always associated with the same bucket, so lookup, create, and delete always happen within the same bucket. This method avoids searching all disks on the CDE. If a bucket has no disks (because of disk failure, unmounting of the disks, and so on), content is served from the network. The incoming traffic to the SE is distributed evenly to the buckets, which means that if the number of available disks in a bucket is less than the other buckets, the other disks in the impaired bucket are used more, which may impact performance.

Bucket Allocation

A hashing algorithm is used to generate a hash of the content URL, on which a calculation is performed to determine the bucket for the content. This ensures content is distributed evenly among all of the buckets.

Content Manager

The Content Manager module keeps track of all the files in CDNFS, and maintains all content popularity information and stores it in a snapshot file. the Content Manager includes the following enhancements:

- Improved the cache content storage:
  - For a platform with physical memory size less than 32 GB(33,554,432 KB), the maximum cached file entries is 20 million and the maximum cached directories 1 million.
  - For a platform with physical memory size more than 32 GB(33,554,432 KB), the maximum cached file entries is 50 million and the maximum cached directories 10 million.
- Increase maximum length of URL to 2048 characters

Note In calculating the maximum length of the URL (2048 characters), an MD5 hash must be considered as part of the overall URL length, therefore the maximum length of the URL should not exceed 2028 characters.

- Continue to manage cache content objects for all protocol engines
- Maintains share memory containing disk related information
- Monitors disk usage periodically and starts eviction when usage exceeds threshold

- Receives updates on disk information based on CMGRSlowScan process, which scans the entire system after every Primary start-time of slowscan. The Primary start-time of slowscan (or Secondary start-time of slowscan) is set in the Devices > Devices > General Settings > Content Management page in the CDSM GUI.
- Receives updates on each disk during start-up from CMgrSnapshotReader.

Note We recommend that any CDE model that has hard-disk drives (HDDs) instead of solid-state drives (SSDs), and is used to stream ABR content, be configured with a maximum of 5 million objects instead of the 20 million objects. This is because HDD-based hardware requires more seek time to access content. The software can handle 20 million objects, but the hard-drive access time impacts the ABR streaming performance. ABR content consists of a large number of small files, which results in a lot of overhead.

For long-tail content (Windows Media Streaming, Flash Media Streaming, Movie Streamer, and progressive download), the maximum number of content objects can be configured with the default of 20 million objects on HDD-based hardware models.

Two of the HDD-based hardware models are the CDE220-2G2 and CDE250-2M0.

Content Types

The Content Manager manages content object types in the following ways:

- Cache content—Maintains file information such as disk path, file size, and priority
- Prefetched content—Maintains prefetched file disk path in memory to manage the number of prefetched assets in the system
- Hybrid content—Handles the same as cache content, maintains file information
- Related content—Maintains information on parent content disk location, aggregated size and hit count

Create

When a file is created (added), FastCAL library updates the Content Manager with the file location, URL, file size, and hit counts.

If the cache-fill rate (creation rate) is much faster than the deletion rate, the Content Manager sets the unwritable flag for that disk. If a protocol engine wants to create content in the system, FastCAL avoids using that disk for the file creation. If all disks are unavailable, the protocol engine performs a bypass or cut-through operation.

The Content Manager sets the disk unwritable flag for the following reasons:

- Disk usage reaches the DiskStopCreate high watermark (98 percent)
- Total cache content objects reaches the ObjCntStopCreate high water mark (105 percent)
- Deletions exceeds 5000 entries

The Content Manager removes the disk unwritable flag when the following occurs:

- Disk usage is below the DiskStartCreate low watermark (95 percent)
- Content object count is below the ObjCntStartCreate low watermark (100 percent)
- Deletion entries drop below 5000

The status of whether the cache content can be stored is displayed in the **show cdnfs** command.

Update

The Content Manager monitors the cached and prefetched content, but not live content. FastCAL updates the content creation time (if created), hit count, file size, and disk path information in the Content Manager when there is a popularity update call from the protocol engine. Whenever there is a cache content popularity update, the Content Manager stores the popularity information and the file path of the content, and computes the popularity (priority) of the content. If the update is a prefetched content popularity update, the Content Manager ignores the message, but continues to monitor the prefetched URL for statistics.

Because of the Web Engine's capability to request bundling, multiple requests can be severed by a single datasource. The datasource keeps track of the requests that are served from it and calls the popularity update at the end of its lifecycle (before eviction of the datasource).

Delete and Eviction

Deletion operating is when a file is deleted through the CLI or by the protocol engine. The FastCAL library deletes the content object from the disk first, then sends a message to the Content Manager to remove the entry for the deleted content object. If the message is lost, the Content Manager deletes the entry by way of the sanity check which runs after Slow Scan (CMGRSlowScan) is finished or through the eviction process.

The Content Manager is involved in evicting content. If the disk usage high-watermark is reached, the Content Manager starts the eviction process by deleting cached content with the lowest priority first. If the protocol engine uses FastCAL to delete the content, FastCAL deletes the content and updates the Content Manager.

Note The disk path is maintained in a hierarchical manner by breaking the down disk path with the directory node (Dir Node) and file node (File Node). If the number of Dir Nodes exceeds the limit (one million), the Content Manager starts evicting files in a similar process to object count eviction.

Priority Calculation

The priority calculation is based on the current hit count, the size of the content object, and the decay of the content object. The popularity of the content decays over a period of time if the content is not accessed.

By default, the Content Manager prefers to keep small content objects over large content objects, because the overhead of fetching a small object is higher than larger objects. However, this preference is configurable in the following ways:

- CDSM GUI: By choosing the Devices > Devices > General Settings > Content Management page, Cache content eviction preferred size drop-down list.
- CLI: By using the cache content eviction-preferred-size {large | small} command.

Deletion Scenarios

There are five scenarios in which the Content Manager removes content:

1. The disk usage exceeds threshold.
2. Content objects exceed the **cache content max_cached_entries** command value.
3. The Cached directories exceed the **cache content max-cached-dirs** command value.
4. The Delivery Service or SE is removed from the CDSM GUI.
5. The disk is removed or marked as "bad."

6. The **clear cache all** command is entered.

The first two can be categorized as priority-based content eviction, the following two can be categorized as top-down tree-structure deletions, and the last one can be categorized as a forking deletion. In all scenarios, the Content Manager removes all entries for the associated content, and deletes all content from storage (with the exception of disk removal).

Disk Usage Exceeds Threshold

The Content Manager keeps track of disk usage. If the disk usage reaches the disk usage high watermark (93 percent), the Content Manager starts the eviction process. When the disk usage reaches the disk usage low watermark (90 percent), the eviction process stops. The eviction process is based on the following criteria:

- Priority of the content on each disk
- Available space on each disk

Content Count Exceeds Maximum Allowed

The default maximum numbers of cached entries and directories depend on the platform. If the maximum is exceeded for max-cached-entries or max-cached-dirs, the Content Manager starts the eviction process.

Delivery Service or SE Removal

If a Delivery Service is removed from the CDSM or an SE is deregistered from a Delivery Service, the Content Manager creates a deletion task and starts deleting all associated cache content. For prefetched content, the Content Manager removes all references, and Acquisition and Distribution handles the content object deletion.

Disk Removal or Disk Marked as Bad

If a disk has gone “bad” and is removed from the system, UNS is notified. UNS internally calls FastCAL, which notifies the Content Manager. The content object information is removed from the Content Manager. The Content Manager also monitors the disk status every three seconds, and if a disk is removed, the Content Manager removes all associated entries for it.

clear cache all Command

If the **clear cache all** command is entered, the Content Manager creates a child process to delete the cache content. The progress of the clear cache all operation is shown in the **show cache** command output.

Addition and Deletion Processes

Content addition stops at 105 percent of the maximum object count or 95 percent of the CDNFS capacity (disk usage). For example, if the maximum number of objects has been configured as 20 million, the VDS-IS starts deleting content if the object count reaches 20 million, but adding content is still allowed. Adding content stops when the maximum number of content objects reaches 21 million(105 percent of 20 million), which allows time for the content deletion process to reduce the number of objects in the VDS-IS to the configured limit. Adding content resumes only after the number of objects is 20 million or less. The same logic applies to disk usage. The deletion process starts when the disk usage reaches 93 percent, adding content stops when the disk usage reaches 98 percent, and adding content resumes only after the disk usage percentage reaches 95 percent or less.

If adding content has been stopped because either the content count reached 105 percent of the limit or the disk usage reached 98 percent of capacity, the unwritable flag is set in the share memory and when the protocol engine calls create, FastCAL library looks into the share memory and denies the creation request. The protocol engine performs a bypass or cut-through operation.

The **show cdnfs usage** command shows the current status of whether the content is able to be cached or not.

The following is sample output from the **show cdnfs usage** command:

```
# show cdnfs usage
Total number of CDNFS entries  :    2522634
Total space                    :     4656.3 GB
Total bytes available          :     4626.0 GB
Total cache size               :        2.4 GB
Total cached entries           :    2522634
Cache-content mgr status       : Cachable
Units: 1KB = 1024B; 1MB = 1024KB; 1GB = 1024MB
```

If the maximum object count is reached, the following is displayed:

```
Cache-content mgr status:  Not cacheable on the following disk(s): [/disk00-06]
[/disk01-06] [/disk02-01]
105% of max obj count reached :        [/disk00-06] [/disk01-06] [/disk02-01]
```

If the disk usage reaches more than 98 percent, the following is displayed:

```
Cache-content mgr status:  Not cacheable on the following disk(s): [/disk01-06]
[/disk02-01]
98% of disk usage reached:       [/disk01-06] [/disk02-01]
```

Starting from Release 3.3, VDS-IS supports content deletion per Delivery Service and per Service Engine by using wildcards. To remove the dynamically cached content in SEs, you need to request for a content deletion task from the CLI, or the CDSM GUI, or using the API

For each URL, the deletion request will be sent to all Service Engines assigned to the Delivery Service by default. It is also possible for the user to select specific Service Engines to delete content on.

If a Delivery Service or content origin is deleted, all of its cached content will be automatically deleted; the user will not need to manually delete contents for a non-existing Delivery Service or content origin.

For more information on content deletion and deletion tasks, see the "Content Deletion Tasks" section on page 8-35.

Eviction Protection

The Content Manager provides configurable eviction protection for small size content and large size content. The Content Manager eviction algorithm is triggered when the disk usage reaches 93 percent or when the cached object count reaches the configured maximum object count. The eviction algorithm assigns a priority number to each content object based on an algorithm similar to the greedy-dual-size-frequency (GDSF) algorithm. The priority number is based on the size and usage of the object. Small objects are given preference over large objects; that is, they are less likely to be deleted.

To protect incoming small objects from being deleted, use the cache content small-file-eviction-protection global configures command. The **cache content small-file-eviction-protection** command allows you to set the maximum content size (500 KB, 1 MB, 2 MB, 4 MB, 10 MB and 20 MB) and the minimum age (5, 10, 15, 30 minutes) of the content object to be protected from deletion. For example, to set the eviction protection for content objects smaller than 20 MB that were ingested in the last 30 minutes, you would enter the following command:

```
#(config) cache content small-file-eviction-protection max-size-20MB min-duration-30min
```

If the content object being cached is smaller than the configured size, it is inserted into a protection table along with the current time stamp. If the difference between the object's time stamp and the current time is greater than the configured time duration, the object is removed from the protection table.

To protect incoming large objects from getting a low priority and being deleted, use the **cache content eviction-protection** global configure command. The **cache content eviction-protection** command allows you to set the minimum content size (100 MB, 500 MB, 1 GB, and 4 GB) and the minimum age (1-4 hours for 100 MB size, 1, 4, 8, or 24 hours for all other sizes) of the content object to be protected from deletion. For example, to set the eviction protection for content objects larger than 100 MB that were ingested in the last two hours, you would enter the following command:

```
#(config) cache content eviction-protection min-size-100MB min-duration-2hrs
```

If the content object being cached is larger than the configured size, it is inserted into a protection table along with the current time stamp. If the difference between the object's time stamp and the current time is greater than the configured time duration, the object is removed from the protection table.

When the eviction algorithm is triggered, before it selects an object for deletion, it first looks at the protection table, and if the object is found, it is skipped for that iteration. The **clear-cache-content** command also checks the protection table before deleting an object. The **clear-cache-all** command does not check the eviction protection table; only the cache content is deleted. As for relative cache content, content in the protection table might still be deleted if the relative content is not protected. The small content eviction protection and large content eviction protection is disabled by default.

If the Content Manager eviction algorithm is not able to find any content to delete, a syslog message is sent to notify the administrator to revisit the configuration. Changing the settings of the **cache content small-file-eviction-protection** or **cache content eviction-protection** command only affects the content that is currently in the protection table and any new content that is added. Any object that is removed from the protection table prior to the configuration change is not brought back into the protection table.

The **no cache content small-file-eviction-protection max-size-xx duration-xx** command removes all small content protection entries in the eviction protection table. The **no cache content eviction-protection min-size-xx duration-xx** command removes all large content protection entries in the eviction protection table. Reloading the SE clear all entries in the eviction protection table.

Web Engine Integration with FastCAL

The Web Engine calls FastCAL directly for content creation, lookup, update, and deletion.

CAL Queue Limits

The CAL queue is limited to 3000 tasks on the CDE250 and 1500 tasks on all other CDEs. When the CAL queue threshold is exceeded, the Web Engine does not add anymore disk operation tasks (creates, updates, or popularity updates) and a trace message is logged with the following string:

```
Reason: CalQThreshold Exceeded!
```

A new output field, “Outstanding Content Popularity Update Requests,” has been added to the **show statistics web-engine detail** command. At any point, the sum of the “Outstanding Content Create Requests,” “Outstanding Content Update Requests,” and “Outstanding Content Popularity Update Requests,” output fields is always less than the threshold. If the sum of these three output fields exceeds the CAL queue threshold, no more create, update, and popularity update tasks are performed, the “Reason: CalQThreshold Exceeded!” trace message is logged, and content is served as follows:

- Large content files are served by way of bypass
- Small content files are served from tmpfs and the files are evicted from tmpfs without moving them to disk

UNS Integration with FastCAL

UNS is the process that is called by other modules like CMS, Acquisition and Distribution, and Streamscheduler to access the CDNFS content by way of the CAL–UNS client library. UNS still handles Movie Streamer and Windows Media Streaming content (both prefetched and cached), and live streaming content for Flash Media Streaming.

UNS uses FastCAL for any disk-based operation. The Content Manager and FastCAL handle accounting of disk usage and new content allocation to the disks for all modules.

Stream and Cache-Fill Performance

The Stream and Cache-Fill feature improves performance in the following ways:

- QoS support. Using QoS together ensures that sessions receive either best effort or a guaranteed rate while not exceeding overall system capacity.
- File-level hole management is supported, allowing partially-filled files to be streamed, and multiple-parallel fills and streams to be attached to the same file at various offsets.
- Higher performance data transmission engine provides better throughput

Hole management is not used for small files, because the sessions are over quickly, and the entire file is always downloaded from the Origin server. Encrypted HLS traffic and HTTPS traffic do not use the Stream and Cache-Fill components, because HTTPS traffic is encrypted in the user space, and encrypted HLS traffic is similar to small ABR files.

With ABR large files, files are either stitched from fragments or they are natively large files. Clients are more likely to stop streaming from an ABR large file when they shift bit rates, so files may have holes.

Hole management and QoS optimize the serving of large ABR files. There is a large improvement in performance with ABR large files and the Stream and Cache-Fill feature.

Large file progressive download traffic is similar to large ABR files, but the client is likely to stay on a bit rate longer because it does not automatically adjust its rate. This traffic type also sees a large performance improvement, for the same reasons as large ABR files.

Stream and Cache-Fill Feature Components

The Stream and Cache-Fill feature consists of the following components:

- QoS Types, page 1-13
- Hole Management, page 1-15

QoS Types

The Stream and Cache-Fill feature adds support for the following QoS classes:

- Hard Guaranteed (HG)—Flows assigned a bit rate that is maintained under any circumstances. The bandwidth allocated for these sessions is never reused by other sessions. HG is not directly selectable.

Note HG is not supported in Release 3.1.

- Soft Guaranteed (SG)—Flows assigned a fixed bit-rate, unlike HG, any unused bandwidth assigned can be reused by other sessions. SG and best effort (BE) flows can continue to be admitted even if the total requested SG rate exceeds system capacity, as long as the total measured rate does not exceed the total system capacity. This is a *statistical guarantee* in the sense that it is expected to be guaranteed in most circumstances.
- Best Effort (BE)—Depending on whether the traffic is VOD or live, the bandwidth allocation behaves differently.
  - In the VOD case, all BE streams are given an equal share of any disk bandwidth left over after guaranteed sessions are satisfied.
  - In the live case, each BE client is allowed to stream at a rate limited only by CPU and network interface bandwidth.

  The system has an administratively-defined minimum best-effort rate for VOD BE sessions. New sessions are only admitted if the global best-effort rate does not fall below the minimum. This way the SE does not stream countless sessions at very low bit-rates.
- On any given SE, live BE traffic cannot be mixed with any other type of QoS, but VOD BE can be mixed with guaranteed QoS types.

Table 1-3 summarizes the different QoS types in the VDS-IS. The types listed in bold are introduced with this feature.

Table 1-3 Supported QoS Types

QoS Type	Minimum Guaranteed Rate	Maximum Rate	Other Compatible QoS Types
Hard Guaranteed (HG) (not supported in Release 3.1)	Protocol Engine requested rate	Protocol Engine requested rate	SG, BE
Soft Guaranteed (SG)	Delivery service bitrate	Delivery service bitrate	HG, BE
Best Effort VOD (BE-VOD)	Globally configured minimum	(Total disk rate - Total (SG + HG) rate) / number BE sessions	SG
Best Effort Live (BE-live)	None	Determined by CPU and network cards	None
Fixed Bit Rate	None	Delivery service bitrate	None
Best Effort	None	Determined by CPU, network cards and disk	None

There is no performance penalty for using any QoS type. The QoS types are defined indirectly through the delivery services.

QoS statistics can be viewed by using the **show statistics admission** command.

Note Only admission statistics can be cleared. QoS statistics are dynamically measured quantities rather than counters; and therefore, cannot be cleared.

Hole Management

Although hole management is not directly visible as a feature to the user, it has a great impact on system behavior. The basic ideas of hole management are as follows:

- Multiple fills can run on a single file at different offsets
- Play request is considered a hit either if the entire request range is filled, or a currently active fill will eventually fill that range
- Maximum number of holes per file is limited for file system robustness reasons

Holes in a file are created in two cases:

1. If the last client aborts the session and fills are still going to the file.
2. When fills are aborted for some other reason like the Origin server drops the connection.

In either case, hole management is equipped to handle these holes by starting fills as needed to bridge holes and limit the total number of holes in a file if required.

When a client aborts a session, any associated fill task normally continues to completion, until either the entire hole is filled or until the end of file. The exception to this is when no more clients are playing the file. In this case, all fills are aborted.

NAS

Network-attached Storage (NAS) is supported as a read-only storage repository at the root location (Content Acquirer) in the VDS-IS. Content is written to NAS by an external agent, such as the Origin Server, a publishing subsystem, or a data storage application. NAS offers a new content category, similar in characteristics to dynamically-cached content, which does not require metadata attachment.

Note NAS is only supported in lab integrations as proof of concept.

The following rules apply to NAS support:

- NAS cannot be used as a source for prefetched or hybrid content.
- Only content serviced by the Web Engine is supported (HTTP content and Flash Media Streaming).

  Note NAS for Windows Media Streaming and Movie Streamer is not supported.

- Only Network File System (NFS) mounts are supported for acquiring content from NAS.
- Content acquired from NAS is not written to local storage on the SEs at the root location; when reading content, NAS is considered an extension of the local file system.
- If there is more than one SE at a root location for a Delivery Service, then the SE that acquires the content from NAS is based on a hash of the content URL (similar to dynamically-cached content).
- NFS share can be mounted from multiple IP addresses simultaneously.
- Multiple mounts for the same volume on a NAS is supported.
- NAS should be collocated with the SEs at the root location; if WAN link is used, then WAN link failover scenario should be provided.
- IP address failover by NAS should be implemented to avoid service disruption.
- NAS is not applicable to live streaming.
- NAS lookup is tried before pulling content from the Origin Server.

- When the Web Engine performs FastCAL lookup, NAS file lookup is performed first; followed by cached content, then prefetched content.
- In a cache-miss scenario, the Origin Server is queried last.

Note Ingress traffic from NAS mounts is not distributed evenly over port channels. Separate interfaces can be used for NAS outside of the port-channel configuration to achieve better load balancing. Ingress traffic to the VDS-IS is determined by the switch, this applies to all application traffic over port channels.

Content Acquirer

Every Delivery Service requires a Content Acquirer, which is a CDA that resides on every Service Engine. The Content Acquirer CDA becomes active when the Service Engine is designated as the Content Acquirer in a Delivery Service. The Content Acquirer has the following functions and capabilities:

- Fetches content from origin servers using HTTP, HTTPS, FTP, or CIFS (Dynamic ingest supports HTTP only).
- Creates and distributes the metadata for each of the prefetched contents according to the Manifest file and the information returned by the origin server.

Once the Content Acquirer has ingested the content and distributed the metadata, it creates a database record for the metadata and marks the content ready for distribution. All other types of ingest (dynamic, hybrid, and live stream) are handled by the Content Acquirer as well.

Starting with Release 3.2.2, when the Content Acquirer sends a request to the Origin Server and when the Content Acquirer distributes the content through multicast, Differentiated Services Code Point (DSCP) marking is done on the outgoing content request to the Origin Server and on the data distributed through multicast to other Internet Streamers.

QoS value for content ingest and **QoS value for multicast data** set in **Delivery Services Definition** page in CDSM GUI are used as DSCP values when the Content Acquirer does content ingest from the Origin Server and when the Content Acquirer does content distribution to other Internet Streamers respectively.

Note Starting with Release 3.3.0, VDS-IS supports per-session DSCP marking for Flash Media Streaming for both VOD and live which is configured differently by Service Rule file.

Internet Streamer

All Internet Streamers participating in a Delivery Service pull the metadata from a peer Internet Streamer called a *forwarder*, which is selected by the internal routing module. Each Internet Streamer participating in a Delivery Service has a forwarder Internet Streamer. The Content Acquirer is the top-most forwarder in the distribution hierarchy. In the case of prefetched ingest, each Internet Streamer in the Delivery Service looks up the metadata record and fetches the content from its forwarder. For live or cached content metadata, only the metadata is distributed.

The content associated with the metadata for live and cached content is fetched by the specified protocol engine, which uses the dynamic ingest mechanism. When a request for a non-prefetched content arrives at an Internet Streamer, the protocol engine application gets the information about the set of upstream Internet Streamers through which the content can be acquired. In the case of dynamic ingest, the Internet Streamer uses the cache routing function to organize itself as a hierarchy of caching proxies and performs a native protocol cache fill. Live stream splitting is used to organize the Internet Streamers into a live streaming hierarchy to split a single incoming live stream to multiple clients. The live stream can originate from external servers or from ingested content. Windows Media Engine, Movie Streamer Engine, and Flash Media Streaming engine support live stream splitting.

Note VDS-IS Release 3.2.2 supports only prepositioned content and does not support Live Stream, Windows Media Engine, Movie Streamer Engine, and Flash Media Streaming because this release is primarily intended for VOD applications.

The Internet Streamers use service control to filter and control incoming requests for content. The Service Rules and Authorization Server with IP address and geographic-location blocking are some of the functions that are encapsulated under the Service Control option in the Internet Streaming CDSM.

The Internet Streamers send keepalive and load information to the Service Router that is participating in the same Delivery Service. This information is used by the Service Router to choose the most appropriate Internet Streamer to handle the request.

Starting with Release 3.2.2, when the receiver sends a NAK packet to the sender for any missed data packets, Differentiated Services Code Point (DSCP) marking is done on the NAK packet sent.

The DSCP value is obtained from the value set for **QoS value for multicast data** set from the **Delivery Services Definition** page in the CDSM GUI is used as DSCP value when the receiver sends NAK packet to the sender.

The Internet Streamer function is implemented as a set of protocol engine applications. The protocol engine applications are as follows:

- Web Engine, page 1-17
- Windows Media Streaming Engine, page 1-22
- Movie Streamer Engine, page 1-26
- Flash Media Streaming Engine, page 1-28

Web Engine

All HTTP client requests that are redirected to a Service Engine by the Service Router are handled by the Web Engine. On receiving the request, the Web Engine uses its best judgment and either handles the request or forwards it to another component within the Service Engine. The Web Engine, using HTTP, can serve the request from locally stored content in the VDS-IS or from any upstream proxy or origin server.

An HTTP client request that reaches the Service Engine can either be from a Service Router redirect or from a direct proxy request.

On receiving an HTTP request for content, the Web Engine decides whether the content needs to be streamed by the Windows Media Engine, and if so, hands the request over to the Windows Media Engine, otherwise the request is handled by the Web Engine. The message size between Web Engine and Windows Media Streaming is 12 KB.

The Web Engine interfaces with the storage function in the Service Engine to determine whether the content is present locally or whether the content needs to be fetched from either an upstream Service Engine or the origin server.

Starting with Release 3.2.2, when the Web Engine requests content from the Origin Server, DSCP marking is done on the outgoing content request from Web Engine to Origin Server. The Web Engine uses the **QoS value for content ingest** value set in **Delivery Services Definition** page in CDSM GUI when it does content ingest from the Origin Server.

Note The Web Engine supports the following:

- Optimization for small content objects
- Optimization of Adaptive Bitrate Streaming for Move, Apple iPhones, and Smooth HD
- Move video on demand (VOD) streaming
- Move live streaming
- MP3 live streaming
- Interoperation with Apple's media stream segmenter, as well as Microsoft's Internet Information Services 7.0 (IIS7.0) Smooth Streaming.
  - Apple's media stream segmenter segments encoded media into separate files for streaming to iPhones.
  - Microsoft's IIS Smooth Streaming offers adaptive streaming of high-definition (HD) content.
- HTTP GET and HEAD request methods.

Bursts of traffic (such as 800 connections per second) may cause the Web Engine to become disabled before it can transmit notification to the SR that the threshold has been reached.

When the content file is smaller than the chunk size, the Unified Kernel Streaming Engine (UKSE) sends the entire file immediately. In this case, the UKSE does not check pacing; therefore, the bit rate for files smaller than the chunk size is not honored.

Cache-Fill Operations

The Web Engine communicates to the upstream Service Engine for cache-fill operations. This interaction is based on HTTP. This cache-fill operation is on demand and therefore only occurs when the content is not stored locally. The upstream Service Engine can be selected dynamically by means of the Hierarchical Cache Routing Module, or can be configured statically through the Internet Streaming CDSM. The Hierarchical Cache Router generates a list of upstream Service Engines that are alive, ready to serve the request, and part of the Delivery Service. If the Web Engine is unsuccessful in locating the content on one of these Service Engines, the content is retrieved from the origin server.

Note When cache-control:no-store is sent in a 200 response from the Origin server, the Web Engine respects the no-store header and does not cache the content. However, if no-store is appended to the cache-control header in a 304 response, the no-store header does not trigger deletion of the content from the disk. The 304 response only triggers updating the cache with the recent header attributes sent in the 304 response header.

The Web Engine supports request headers and entity headers as described in the HTTP 1.1 specification (RFC 2616). The Web Engine allows VDS-IS domain and HCACHE custom headers only when sent from an SE.

Web Engine respects the following date formats:

- Sun, 06 Nov 1994 08:49:37 GMT ; RFC 822, updated by RFC 1123
- Sun Nov 6 08:49:37 1994 ; ANSI C's asctime() format

The following format is obsolete and is not supported:

- Sunday, 06-Nov-94 08:49:37 GMT ; RFC 850, obsoleted by RFC 1036

If the headers (for example, the expiry header) are received with a non-supported date format, the Web Engine continues to cache the content, but subsequent requests for the same URL are revalidated as the content is considered expired.

Whether the content is found locally or retrieved and stored through the cache-fill operation, the Web Engine serves the content based on the following:

- **Freshness of content**—The freshness of prefetched content is governed by a Time to Live (TTL) value set for the content in the Delivery Service configuration. The TTL specifies the rate at which content freshness is checked. This setting is configured for each Delivery Service either by using the CDSM or by specifying this setting in the Manifest file for the Delivery Service.

  For cached content, which is content ingested by means of the dynamic ingest or the hybrid ingest method, the freshness check is performed by the Web Engine in compliance with RFC 2616. If the origin server does not provide an expiry time, the Web Engine uses the age multiplier setting, the minimum TTL setting, and the maximum TTL setting to determine the freshness of the content. If the Web Engine performs the calculation and determines that the content should be checked for freshness with the origin server, and the origin server is unreachable, the client receives a 504 error.

Note This algorithm is used to determine freshness for cached content based on the expire time. It is not used to determine the popularity of the content.

This expiry header validation is just one case used to decide whether content revalidation is needed or not. Revalidation is also decided based on cache control headers that are part of request headers, and the min-fresh, max-stale, max-age parameters that can come in both request and response headers.

Revalidation is enabled by default for the Web Engine.

  If the origin server provides the expire time, it is used to determine the freshness of the content. If the expire time is not available, the expire time of the content is calculated as follows:

  Expire_time = (*Create_time – Last_modified_time_from_origin_server*) * *age multiplier*

  The *create time* is the time on the VDS-IS when the content was cached. The *last modified time* is the time the content was last modified on the origin server. The *age multiplier* value (as a percentage) is used to shorten the time that it takes to have the content revalidated.

  For example, if the create time was May 5, 2009 12:00 and the origin server last modified the content on May 1, 2009 12:00, then the expire time would be 4 days. If the age multiplier was set to 50 percent, the expire time would be 2 days.

  The calculated expire time is compared with the minimum TTL and maximum TTL settings. If the expire time is greater than the maximum TTL, the maximum TTL is used as the expire time. If the expire time is less than the minimum TTL, the minimum TTL is used as the expire time.

  Using the example above, if the minimum TTL was 3 days and the calculated expire time was 2 days, then the minimum TTL is used as the expire time. If the maximum TTL is 10 days, then the calculated expire time still uses the minimum TTL of 3 days as the expire time. The min/max TTL algorithm follows:

```
Expire_time =   if (MINTTL < Expire_time < MAXTTL), then Expire_time
                  else if Expire_time < MINTTL, then MINTTL
                  else MAXTTL
```

The expire time is compared with the *cache age* to determine whether the content needs to be revalidated by the origin server. If the cache age is less than or equal to the expire time, then the content is considered fresh. The following calculation is used to determine the cache age:

Cache_age = Current_time – Create_time

In our example, if the current time is May 25, 2009 12:00 and the create time is May 5, 2009 12:00, then the cache age is 20 days. The cache age of 20 days is compared to the expire time, which in our example is 2 days, and because the cache age is greater than the expire time the content is revalidated with the origin server. When the content is revalidated it gets a new create time. To compute a more accurate cache age, the response delay is considered. The *response delay* is calculated as follows:

Response_delay = Create_time – Time_request_sent_to_origin_server

In our example, the create time is May 5, 2009 12:00, and if the origin server takes 2 minutes to respond to the request for content (because of network-imposed delays), the response delay is May 5, 2009 11:58. This allows the cache age to be calculated based on the time the request was initiated, not the time the response was received.

- **Rate of data transfer**—The rate at which the content is sent can be configured on a per-delivery basis. By default, LAN bandwidth is used.
- **Content completeness**—Prefetched content is stored locally in the VDS-IS in its entirety. For cached content, there are two cases when the content is not complete:
  - The Web Engine process halts or the Service Engine experiences a failure in the process of caching the content. In this case, the subsequent request starts the cache fill anew.
  - The content is in the process of being cached by another request. In this case, the subsequent request is served from the cached content.

Dynamic Caching

Starting with Release 3.2.2, dynamic caching is configurable per Delivery Service. By default, dynamic caching is enabled. If a content requested by the client is not present in the cache then the Web Engine sends a request to an upstream streamer to acquire the contents, caches the contents and then delivers it to the client.

By making the dynamic caching configurable at Delivery Service level, the user has the option of disabling the lookup from the origin server.

If the requested content is not available in a particular streamer or service engine, and dynamic caching is disabled, the client receives a 403 error response (HTTP FORBIDDEN).

The dynamic caching feature is configured in the Services > Service Definition > Delivery Services > General Settings page in the CDSM GUI.

If dynamic caching is disabled, only prepositioned content and contents cached before dynamic caching was disabled for which cache revalidation is not required will be served to the client.

The dynamic cache setting for a given Delivery Service will override the following configuration properties:

- The web-engine cache settings for age-multiplier, max-ttl, min-ttl will not be affected by dynamic caching configuration.
- The web-engine revalidation setting will be overridden by the dynamic cache setting for a given Delivery Service. No cache revalidation will be done if dynamic caching is disabled for the contents pertaining to that Delivery Service. If the complete content is available, it is served without any revalidation.
- The cache bypass requests will not be processed if dynamic caching is disabled.

- If dynamic caching is disabled and only partial content is available, then client receives a 403 error message.

Per-Request HTTP Headers from Redirected URLs

When the VDS-IS is integrated with products such as the Content Adaptation Engine (CAE), Service Engines are used to serve over-the-top content. In such scenarios, the VDS-IS provides HTTP headers similar to that of the Origin server. The information in the HTTP header can be unique to a user session.

The CAE retrieves this information from the Origin server response and provides it as a query parameter within the URL. This information is intact when received by the Service Engine following redirections from the CAE and Service Router. The Web Engine retrieves the "_resp_hdrs_" value from the received URL. The retrieved value is % unescaped, and parsed for use when serving the content.

As indicated in RFC 2396, a query parameter cannot contain the reserved characters ;/?:@&=+,$ and thus are escaped using % encoding. The query string must have the "_resp_hdrs_" tag. A URL with the "_resp_hdrs_" tag has the following format:

http://<URL to serve>?_resp_hdrs_=<strings to include in the http headers>

The following is an example of a URL with the "_resp_hdrs_" tag and value:

http://nas_url_to_serve?_resp_hdrs_=Set-Cookie%3A%20ff%3DrlsBo4v%3B%20path%3D/%3B%20
domain%3D.site.com

HTTP Error Response Caching

Caching HTTP error responses from the Origin Server provides the Web Engine with the ability to validate incoming requests faster and reduce unnecessary access to the Origin Server.

As an example, the Origin Server sends back a response with the status "503 Service Unavailable" and includes the *maximum age* in the response. The Web Engine caches the response locally, and for any subsequent client requests for the same content, the Web Engine compares the cached response age with the maximum age returned in the response. If the cached response is expired, the Web Engine rechecks the Origin Server; otherwise, the Web Engine sends the cached response to the client.

The HTTP response headers must include the max-age, expiry, etag, and other fields that are required to determine whether the responses can be cached.The HTTP response headers that can be cached are those that indicate some error has occurred with respect to the client request (4xx or 5xx status codes).

Note Error response 416 is not cached when the Origin server responds with Transfer-Encoding:Chunked header. Whenever the Origin server sends chunked encoding, whatever status is returned, the response is not cached.

Service Rules

Service rules can be configured that dictate how the Web Engine responds when client requests match specific patterns. The patterns can be a domain or host name, certain header information, the request source IP address, or a Uniform Resource Identifier (URI). Some of the possible responding actions are to allow or block the request, generate or validate the URL signature, or rewrite or redirect the URL.

Note The following Service Rule actions are supported for the Web Engine: allow, block, rewrite the URL, no cache, redirect the URL, resolve the URL, revalidates cache, and validate the URL signature.

Windows Media Streaming Engine

The Windows Media Streaming engine uses Windows Media Technology (WMT), a set of streaming solutions for creating, distributing, and playing back digital media files across the Internet. WMT includes the following applications:

- Windows Media Player—End-user application
- Windows Media Server—Server and distribution application
- Windows Media Encoder—Encodes media files for distribution
- Windows Media Codec—Compression algorithm applied to live and on-demand content
- Windows Media Rights Manager (WMRM)—Encrypts content and manages user privileges

The Windows Media Streaming engine streams Windows Media content, with the capability of acting both as a server and as a proxy. It streams prefetched content to the Windows Media Player, acts as a proxy for client requests, splits a live stream into multiple live streams, and caches content requested from remote servers.

Windows Media Streaming engine acts as Windows Media Server for prefetched or cached content stored locally. The request is served by RTSP and HTTP. Windows Media Streaming engine checks with the storage function on the Service Engine to see whether the content is stored locally; if the content is not found, Windows Media Streaming engages the Windows Media Proxy.

The WMT Proxy works like the cache-fill operation in the Web Engine. See the "Cache-Fill Operations" section on page 1-18. There are two options:

- Hierarchical Caching Proxy—If content is not found locally, Windows Media Streaming checks the upstream Service Engines first before pulling the content from the origin server.
- Static Caching Proxy—The administrator statically configures Service Engines as upstream proxies.

The WMT Proxy accepts and serves streaming requests over RTSP and HTTP.

For information on cache management for Windows Media Streaming, see the "Content Manager" section on page 1-7.

Fast Start

Fast Start provides data directly to the Windows Media Player buffer at speeds higher than the bit rate of the requested content. After the buffer is filled, prefetched, cached, or live content stream at the bit rate defined by the content stream format. Fast Start does not apply to content that is dynamically ingested. Only Windows Media 9 Players that connect to unicast streams using MMS-over-HTTP or RTSP can use Fast Start. The Fast Start feature is used only by clients that connect to a unicast stream. With live content, Windows Media Streaming needs to hold the content in its buffer for a few seconds. This buffer is used to serve Fast Start packets to subsequent clients that request the same stream as the initiating first client request. The first client triggers the process, with the subsequent clients benefiting from Fast Start.

Fast Cache

Fast Cache allows clients to buffer a much larger portion of the content before rendering it. Fast Cache is supported only for TCP. Windows Media Streaming streams content at a much higher data rate than specified by the stream format. For example, using Fast Cache, Windows Media Streaming can transmit a 128-kilobit per second (Kbps) stream at 700 Kbps. This allows the client to handle variable network conditions without perceptible impact on playback quality. Only MMS-over-HTTP and RTSP requests for prefetched or cached content support Fast Cache. The speed is determined by the client's maximum rate and the configured Fast Cache rate—whichever is smaller.

Fast Stream Start

The first client requesting a live stream often experiences the longest wait time for the content to begin playing. Users can experience long wait times because of the full RTSP or HTTP negotiation that is required to pull the live stream from the source. Delays can also occur if the edge Service Engine has not buffered enough stream data to fill the player’s buffer at the time the content is requested. When the buffer is not filled, some data to the client might be sent at the linear stream rate, rather than at the Fast Start rate. With Fast Stream Start, when a live stream is primed, or scheduled and pulled, a live unicast-out stream is pulled from the origin server to a Service Engine before a client ever requests the stream. When the first request for the stream goes out, the stream is already in the Delivery Service.

Caching SDP Files for RTSP Broadcast Live

Live streaming is content that is streamed while it is still being encoded by an encoder. The two kinds of Windows Media live streaming are as follows:

- Playlist live—One or more content items are streamed sequentially.
- Broadcast live—Live and prerecorded content can be streamed to more than one client simultaneously. The SE streams the content to all clients, which does not allow the clients to perform seeks on the stream.

Streaming is accomplished by using HTTP live or RTSP live. HTTP live uses Windows Media Streaming Protocol (MS-WMSP) where the wms-hdr in the WMS-Describe-Response describes the content. RTSP live uses RTSP where the Session Description Protocol (SDP) file in the DESCRIBE response describes the content.

The RTSP playlist live SDP file cannot be cached because the SDP file keeps changing to reflect the different content playlists.

The SDP file for RTSP broadcast live does not change unless the program is stopped, so it can be cached on the streaming SE. Once the SDP file is cached, it can be used to compose the DESCRIBE response. No further requests for the SDP file from the upstream server (SE, Content Acquirer, or Origin server) are necessary.

Note The SDP file cannot be cached if content requires authorization by either the Origin server or the SE.

Live Stream Splitting

Live stream splitting is a process whereby a single live stream from the origin server is split and shared across multiple streams, each serving a client that requested the stream. When the first client that requested the stream disconnects, Windows Media Streaming continues to serve the subsequent requesting clients until all requesting clients have disconnected. Live stream splitting using content that is already stored locally is generally better than using content from the origin server; this is because the Service Engine is typically closer to the requesting clients, and therefore network bandwidth to the origin server is freed up.

To avoid doing a CAL lookup resolve for each incoming Windows Media Streaming live request, the live hierarchical splitting URL is cached and is then used by all subsequent Windows Media Streaming live requests for the same live program.

Note When using Windows Media Server 2008 as the origin server, the source content type must be a playlist or encoder type.

Live stream splitting can either be unicast or multicast, depending on the configuration, capabilities and limitations of the network. Windows Media Streaming can receive and deliver Windows Media content over IP multicast or unicast transmission in the following combinations:

- Unicast-In Multicast-Out
- Multicast-In Multicast-Out
- Unicast-In Unicast-Out
- Multicast-In Unicast-Out

Note For multicast-in (to the SE) to work, the network needs to be multicast-enabled.

Multicast-Out

Windows Media Streaming can be used in a live or rebroadcast program to deliver multicast streams to client devices. The source of the stream can be multicast, unicast, or a local file. The program can be scheduled, continuous, or play once. The content can be either live or rebroadcast. Windows Media Streaming creates a Windows Media file (.nsc) that contains session information including the multicast IP address, port, Time to Live (TTL), and so on. The client requests the .nsc file using HTTP. Once the file is downloaded, the client parses it and sends an Internet Group Management Protocol (IGMP) join to receive the multicast stream. A client can start and stop the stream, but cannot pause, fast-forward, or rewind it.

Unicast-Out

Windows Media Streaming can act as a broadcast publishing point to deliver live streams, prefetched/cached content, or content from dynamic ingest, to a requesting client. The source of the stream can be multicast, unicast, or a local file. Windows Media Streaming can also perform live stream splitting if more than one client requests the same content. The Delivery Service can be used to simulate an experience similar to viewing a TV program even if the source of the stream is a Video on Demand (VOD) file. A client can start and stop the stream but cannot pause, fast-forward, or rewind it. When a Delivery Service is configured, a client makes a request to the Windows Media Engine, which is acting as the Windows Media Server, and Windows Media Streaming checks to see whether the incoming stream is present. If it is, Windows Media Streaming joins the stream and splits it to the new client. If the request is the first client request for this stream, Windows Media Streaming sends the request to the origin server and then serves it to the new client.

ASX Request Handling

Web Engine generates meta-responses for the following Windows Media Streaming ASX requests:

- Requested Windows Media Streaming asset is prefetched
- Unicast (.asx) request for Windows Media Streaming live program is scheduled
- Multicast (.nsc.asx) request for live program is scheduled

When the **wmt disallowed-client-protocols** command is configured, Web Engine generates the meta-response based on the protocols enabled. When both RTSPU and RTSPT are disabled, only the HTTP URL is generated in the meta-response. However, when HTTP is disabled, the generated ASX file still contains the HTTP URL, so that the content can be served by the Web Engine as a progressive download (as opposed to live streaming by Windows Media Streaming). For .nsc.asx files, only the HTTP URL is generated.

VOD ASX Request

Web Engine does lookups for incoming ASX requests in the following manner:

- If the ASX asset is cached or prefetched, the asset is served.
- If the ASX asset is not found, Web Engine strips the .asx extension from the URL and performs the lookup again.
  - If the asset is found (after stripping the .asx from the URL), Web Engine generates the meta-response for the requested Windows Media Streaming ASX request.
  - If the asset is not found (after stripping the .asx from the URL), no meta-response is generated and the request is treated as a cache miss.

Live ASX Request

In the case of a unicast live request or a multicast live request, the Web Engine generates the meta-response for found assets. If the asset is not found, Web Engine generates a "403 Forbidden" error message and sends it to the client.

NSC Request Handling

An NSC request is a managed live streaming request. When the live program schedule is started, the NSC content is created by Windows Media Streaming.

For Web Engine lookups of NSC files, CAL returns the NSC file location where the content can be served from, or returns "Not in Schedule."

Authentication

Windows Media Streaming supports pass-through authentication. The following authentication mechanisms are supported in pass-through mode:

- Anonymous
- NTLM
- Negotiate (Kerberos)
- Digest access authentication

With pass-through authentication, Windows Media Streaming establishes a tunnel between the client and the origin server so that the origin server can authenticate the client.

Bandwidth Management

Bandwidth management of Windows Media content can be controlled by setting limits for incoming and outgoing bandwidth and session bit rate and Fast Start maximum bandwidth. In addition, in the case of live streaming, contributing origin servers can by identified to allow incoming content to exceed the bandwidth check to support high demand scenarios. The Windows Media bandwidth management capabilities are described in Table 1-4.

Table 1-4 Bandwidth Management Capabilities

Bandwidth Management	Description
Incoming Bandwidth	The bandwidth for Windows Media content coming into the Service Engine, from either an upstream Service Engine or from the origin server.
Outgoing Bandwidth	The bandwidth for streaming Windows Media content to the end user from the Service Engine.
Incoming Session Bit Rate	The maximum bit rate per session that can be delivered to the Service Engine from the origin server or upstream Service Engine.

Table 1-4 Bandwidth Management Capabilities (continued)

Bandwidth Management	Description
Outgoing Session Bit Rate	The maximum bit rate per session that can be delivered to a client.
Incoming Bandwidth Bypass List	The list of identified hosts allowed to bypass the incoming bandwidth check for broadcast or multicast live content.
Fast Start Maximum Bandwidth	Maximum bandwidth allowed per player when Fast Start is used to serve packets to each player. Increased bandwidth initially used by the Fast Start feature can overburden a network if many players connect to the stream at the same time. To reduce the risk of network congestion caused by the Fast Start feature, limit the amount of bandwidth the Fast Start feature uses to stream to each player.

Movie Streamer Engine

The Movie Streamer Engine is an open-source, standards-based, streaming server that delivers hinted MPEG-4, hinted 3GP, and hinted MOV files to clients over the Internet and mobile networks using the industry-standard RTP and RTSP. Hinted files contain hint tracks, which store packetization information that tell the streaming server how to package content for streaming.

The Movie Streamer Engine is an RTSP streaming engine that supports Third Generation Partnership Project (3GPP) streaming files (.3gp). Support of 3GPP provides for the rich multimedia content over broadband mobile networks to multimedia-enabled cellular phones.

Note The streaming capability of Movie Streamer Engine only depends on the movie file format or stream transport type. It is independent of codec types. Movie Streamer supports any client player that can fetch media streams by way of RTSP or RTP. However, the client player must have the correct codec to render the stream correctly.

The Movie Streamer Engine can act as both a server and a proxy. It streams prefetched or RTSP-cached content to RTSP clients, acts as a proxy for client requests, splits a live stream into multiple live streams, and caches content requested from remote servers.

After the RTSP request comes into the Movie Streamer, the URI in the RTSP request is modified to reflect the result of the mobile capability exchange. The Movie Streamer checks with the storage function on the Service Engine to see whether the content is stored locally. If the content is not found or if an RTSP-cached content version needs freshness validation, the Movie Streamer engages the Movie Streamer proxy.

In the case of an RTSP-cached content version verification, the Movie Streamer proxy forwards the DESCRIBE request to the origin server for a response containing the Last-Modified-Time header in the response. If the Last-Modified-Time matches the cached version, the Movie Streamer streams the cached content; otherwise, the Movie Streamer proxy forwards the request to the origin server for RTSP negotiation. Then, a client session and a server session are created.

- Server session is responsible for connecting to the origin server to fetch the content and cache it locally. The server session generates the media cache file and the linear hint files.
- Client session is responsible for streaming the locally cached file to the client.

- Client and server sessions are separated so that multiple server sessions can be spawned for the same URL to cache content from different starting points or at faster speeds, or both. This increases the speed of fetching the content. The client session starts to stream from the cached content that the server session is writing.

The Movie Streamer proxy works like the cache-fill operation in the Web Engine and the Windows Media Engine, except for the minimum TTL value. The Movie Streamer's minimum TTL value is always zero. See the "Cache-Fill Operations" section on page 1-18. There are two options:

- Hierarchical Caching Proxy—If content is not found locally, the Movie Streamer checks the upstream Service Engines first before pulling the content from origin server.
- Static Caching Proxy—The administrator statically configures Service Engines as upstream proxies.

For information on cache management for the Movie Streamer, see the "Content Manager" section on page 1-7.

The Movie Streamer supports basic pass-through proxy mode for certain conditions where caching cannot be performed. Such conditions include, but are not limited to, the Service Engine running out of disk space.

Transport Types

Prefetched content can be delivered by the non-accelerated method or the accelerated method. Non-prefetched content (proxied or cached content) is always delivered by the accelerated method. The content is delivered to the client device by one of the following mechanisms:

- **Non-Accelerated**—This method has limited concurrent streams and total throughput, but supports many transport formats. The non-accelerated method supports the following transport formats:
  - RTP over UDP
  - Reliable UDP
- **Accelerated**—This method supports only RTP over UDP. Content must be reprocessed by the Movie Streamer Linear Hinter. The linear hinter process can be initiated manually by the administrator or dynamically triggered by the first request for the content.

The Movie Streamer Linear Hinter process may take a while, so the first request that triggers this process is served by the non-accelerated method. All subsequent requests are served by the accelerated method.

The first client request for content that requires proxying or caching experiences a delay, because all proxying and caching requires the accelerated method.

Live Stream

The Movie Streamer Engine supports multicast reference URLs (Announce URLs) for programs that are created through the Internet Streaming CDSM. The multicast reference URL, which is in the form of http://*Service Engine IP address*/*Program ID*.sdp, is resolved by the Movie Streamers that are serving the live program.

QuickTime live typically has a UDP socket pair (for RTP and RTCP) per track, and each client session typically has two tracks (audio and video).

Note The following rules apply to live splitting:

1. For unicast streaming, the client request must be sent by RTSP.
2. For multicast streaming, the client request must be sent by HTTP.

Authentication

The Movie Streamer Engine supports the Basic authentication mode.

URL Signing

For more information see the "URL Signing" section on page 1-30.

Flash Media Streaming Engine

The Flash Media Streaming engine incorporates the Adobe Flash Media Server technology into the VDS-IS platform. The Flash Media Streaming engine is capable of hosting Flash Media Server applications that are developed using ActionScripts, such as VOD (prefetched content, or dynamic or hybrid ingested content), live streaming, and interactive applications.

Note Starting with Release 4.0, the Flash Media Server 3.5 is upgraded to Adobe Media Server 5.0.2.

The Flash Media Streaming engine supports the Adobe Flash Media Rights Management Server (FMRMS) for VOD content; it is not supported for live streaming. Adobe FMRMS protects media content delivered to Adobe Media Player and Adobe AIR applications. FMRMS is also available for proxied content, if Adobe supports the content type. For more information about the Adobe Flash Media Rights Management Server, see www.adobe.com.

Note VDS-IS supports the Adobe Flash Media Server Administration APIs and the Administration Console that was built using the Administration APIs. These APIs can be used to monitor and manage the Adobe Flash Media Server running on a Cisco VDS-IS Service Engine. See the "Configuring Flash Media Streaming—General Settings," page 4-48 for more information.

Upon receiving a client request for VOD content, the edge Service Engine does the following:

- If the content is present, the edge Service Engine streams it using RTMP.
- If the content is not present, the edge Service Engine uses HTTP to fetch the content from the origin server and serves it using RTMP.

No client information is sent to the origin server. No per-client control connection is present between the edge Service Engine and the origin server for VOD streaming.

HTTP Requests

Flash Media Streaming encompasses all flash applications, from simple Flash Video (FLV) files to more complex Small Web Format (SWF) files. All HTTP client requests for SWF files, that are redirected to a Service Engine by the Service Router, are handled by the Web Engine. The Web Engine, using HTTP, serves the request from locally stored content in the VDS-IS or from any upstream Service Engine or origin server. See the "Web Engine" section on page 1-17 for more information.

RTMP Requests

The SWF file is a compiled application that runs on the Adobe Flash Player, and may contain Real Time Media Protocol (RTMP) calls to FLV, MPEG-4 (H.264), or MP3 files. RTMP calls, in the form of URL requests, are routed to a Service Engine by the Service Router.

Flash Media Streaming supports RTMP and RTMPE on port 1935 only. RTMPE is the secure flash streaming technology from Adobe. Encrypted RTMP (RTMPE) is enabled on Flash Media Streaming by default, and allows you to send streams over an encrypted connection without requiring certificate management.

Flash Media Streaming also supports RTMPT and RTMPTE on port 80. RTMP Tunneled (RTMPT) encapsulates the RTMP data within HTTP requests to traverse firewalls. RTMP Tunneled Encrypted (RTMPTE) encrypts the communication channel, tunneling over HTTP.

Note The Service Router uses RTMP redirection to direct the client's Flash Player to the best Service Engine based on load balancing and resiliency. RTMP redirections are supported only by Adobe Flash Player 9. All older Flash Players do not support RTMP redirection.

Note For VOD streams, all RTMP calls in the SWF file must be in the following format:

```
rtmp://rfqdn/vod/path/foo.flv
```

In this format, *rfqdn* is the routing domain name of the Service Router, *vod* is the required directory, and *path* is the directory path to the content file that conforms to the standard URL specification.

If you are unable to store the VOD content in the required *vod* directory on your origin server, you can create a VOD virtual path for all RTMP requests. All client requests for RTMP calls still use the rtmp://rfqdn/vod/path/foo.flv format for VOD streams, but the SE replaces the *vod* directory with the string specified in the **flash-media-streaming application-virtual-path vod map** command.

Use the **flash-media-streaming application-virtual-path vod map** *mapping string* command on each SE participating in a Flash Media Streaming Delivery Service. The mapping string variable accepts all alphanumeric characters and the slash (/) character, and can be from 1 to 128 characters. For example, to map the "vod" directory to "media" for the go-tv-stream.com origin server, use the **flash-media-streaming application-virtual-path vod map media** command.

If comedy.flv is the content being requested, the RTMP call in the SWF file would be rtmp://go-tv-stream.com/vod/comedy.flv. The SE would replace the "vod" directory and request http://go-tv-stream.com/media/comedy.flv from the upstream SE or origin server.

If just the slash (/) character is used to replace the "vod" directory, the SE request would be http://go-tv-stream.com/comedy.flv.

For prefetched and cached content, the Flash Media Streaming engine uses RTMP or RTMPE over port 1935. The Flash Media Streaming engine also supports RTMPT and RTMPTE over port 80. For content that is not found locally, the Flash Media Streaming engine communicates with the Web Engine, that in turn communicates with the upstream Service Engine for cache-fill operations. See the "Cache-Fill Operations" section on page 1-18. This interaction uses HTTP. Once the content is in the process of being retrieved by the Web Engine, the Flash Media Streaming engine uses RTMP to begin streaming the content.

The following describes the characteristics of caching content using HTTP for RTMP client requests;

1. Origin server-based cache validation is still honored for the cached content.
2. Client-side Web Engine rules are bypassed for the RTMP client request.
3. If HTTP headers from the origin server have the "no-cache" attribute set, content is not cached, and transparent proxy is performed to stream RTMP.

4. Transparent proxy from HTTP to RTMP is supported. Flash Media Streaming engine begins RTMP streaming while content is still being fetched using HTTP proxy mode.

Any HTTP configuration that prevents content from being cached still applies for RTMP requests. The Flash Media Streaming engine uses multiple HTTP-based range requests in such cases.

Multi-Bit Rate Streaming

Flash Media Streaming supports multi-bit rate streaming, also known as dynamic streaming. Dynamic streaming offers the ability to adjust the bit rate used to stream video to clients to adapt to changes in network conditions.

Multi-bit rate streaming has the following requirements:

- The origin server must be running Flash Media Server 3.5
- The client must be using Flash Media Player 10 or later
- The encoder for VOD must be running Flash Media Encoder CS4
- The encoder for live streaming must be running Flash Media Live Encoder 3

For VOD, the encoder creates different bit rates for the content. For live streaming, the encoder publishes three streams with different bit rates to the origin server.

With Flash Media Player 10, there are new QoS properties that provide information about the stream and video performance and network capabilities; for example, when the NetStreamInfoBytesPerSecond field changes, the client can request a different bit rate for the stream.

The client player sends the command to switch or swap the stream. When network changes occur, the client sends a switch command to request the content be streamed with a higher or lower bit rate. Swap is used when swapping streams in a playlist (for example, advertisements). The bit rate change request works for both VOD and live streaming. The supported formats are H.264 and FLV. The client-side ActionScripts should use play2() instead of play() for smooth stream transitions.

Flash Media Streaming Proxy

The Flash Media Streaming engine can deliver content acting as an origin server or as a proxy server. The Flash Media Streaming engine acts as a proxy server when content cannot be cached due to the origin server's configuration or due to the Service Engine's Web Engine configuration. Content is ingested and distributed using HTTP, whether the client request for the content used HTTP or RTMP.

Note Any content that does not contain "live" or "vod" in the path is automatically proxied.

Unicast Streaming

The Flash Media Streaming engine supports unicast flash streaming.

URL Signing

Flash Media Streaming supports signed URLs, which adds additional security. The URL signature generation is based on a key that is a shared secret between the component generating the URL signature and the component validating the URL signature. The URL signature can be generated by the Service Engine, another component external to the Service Engine, or the web portal.

For more information about the URL signatures, see the "Configuring URL Signing Key" section on page 4-28.

Codecs

Flash Media Streaming supports the On2 VP6 codec, as well as those listed in Table 1-5.

Table 1-5 Codecs Supported in Flash Media Streaming

Standard	Details
ISO/IEC 14496-3	MPEG-4 Part 3, also known as AAC+, HE-AAC. A set of compression codecs for perpetual coding of audio signals, including some variations of Advanced Audio Coding (AAC), as well as AAC Main, AAC LC, and SBR.
ISO/IEC 14496-10	Advanced Video Coding (AVC), also known as H.264/AVC. All levels of applications are supported, Base (BP), Main (MP), High (HiP), High 10 (Hi10P), and High 4:2:2 Profile (Hi422P). This standard is technically identical to the ITU-T H.264 standard.
ISO/IEC 14496-12	ISO Base Media File Format. A file format for storing media content containing one audio track (either ISO/IEC 14496-3 [AACPlus] or MP3), and one video track (either ISO/IEC 14496-10 [H.264 or AVC] or VP6).
3GPP TS 26.245	Time text format.

Flash Media Streaming DCSP Marking

Starting with Release 3.3.0, VDS-IS supports per session DSCP marking for Flash Media Streaming including both VOD and Live.

The DSCP value is a 6-bit field in the IP header, which takes any value between 0 and 63. The Delivery Service specific DSCP value shall be set using the AuthSvr Service Rule file. A new XML tag is added in the rules XML file

```
<Rule_Dscp matchGroup="grp1" protocol="rtmp" dscp-bits="10"/>
```

The above rule will match the *matchGroup* defined by a regex pattern or domain name and the attribute *dscp-bits* will be applied to the matching pattern. The attribute is the DSCP value ranging from 0 to 63. If the *dscp bits* is not specified in the rules xml file, the default DSCP value i.e., 0 is considered.

Using rule files provides flexibility to apply DSCP values to different matched patterns such as domain name, URL, IP address, and so on. To support DSCP per Delivery Service, you need to configure the Delivery Service domain name in the rule file.

Note The FMS per session DSCP marking feature is supported only on IPv4 protocol. The feature is disabled for IPv6 protocol by default.

Live Streaming

Flash Media Streaming uses RTMP to stream live content by dynamic proxy. Configuration of live or rebroadcast programs is not required. When the first client requests live streaming content, the stream is created. There are no limits to the number of live streams other than the system load. Live streaming uses distributed content routing to distribute streams across multiple Service Engines.

Upon receiving a client request for live content, the edge Service Engine does the following:

- If the live stream is already present, the edge Service Engine attaches the new client to the existing stream. No message is sent to the origin server and no connection is set up.
- If the live stream is not present, VDS-IS creates a connection to the origin server to get the stream. No client information is sent to the origin server.

No per-client control connection is present between the edge Service Engine and the origin server for live streaming.

For Flash Media Streaming, a Delivery Service can be used for prefetched content, cached content, dynamically cached content, and live content. Because Flash Media Streaming uses dynamic proxy to stream live content, no disk space is used to store content. A Service Engine can act as the origin server for streaming live content, provided the SE designated as the origin server is not assigned to the Delivery Service that is streaming the live content.

The Flash Media Streaming engine automatically retries a connection to an upstream Service Engine or the origin server if the upstream live-splitting connection fails. This switchover does not require any additional retries from the client side. Clients see a subsecond buffering, after which video continues to play. This feature does not address switchover when the Service Engine that is streaming to the client fails. The primary advantage is increased resiliency in the VDS-IS infrastructure. In other words, if a Service Engine fails, the downstream Service Engine automatically tries to connect to an upstream Service Engine in the path, and if it fails to connect, then a connection to the origin server is automatically made.

The Adobe Flash Media Encoder can publish the streams to any Adobe Flash Media Server acting as the origin server. Clients use the RFQDN to get the live content. The request from the client for "streamname" is mapped to origin_appinst_streamname internally in the VDS-IS to differentiate between two streams with the same name in two different delivery services.

Note All RTMP calls for live content in the SWF file must be in the following format:

```
rtmp://rfqdn/live/path/foo.flv
```

In this format, *rfqdn* is the routing domain name of the Service Router, *live* is the required directory, and *path* is the directory path to the content file that conforms to the standard URL specification.

Flash Media Streaming supports live stream splitting. For more information about live stream splitting, see the "Live Stream Splitting" section on page 1-23.

Flash Media Streaming Query String

Previously, if an RTMP request had a query string in the URL for VOD, the Web Engine could decide whether or not to cache the content based on the Web Engine configuration. However, if the query string in the RTMP URL included the end-user specific parameters and not the stream name, every request would have a different URL because every user has a different query string. This leads to the same content getting cached multiple times.

The **flash-media-streaming ignore-query-string enable** command tells Flash Media Streaming to remove the query string before forwarding the request to the Web Engine in the case of VOD, or before forwarding the request to the forwarder SE in the case of live streaming.

If URL signature verification is required, the sign verification is performed before the query string check is invoked. The URL signing and validation, which adds its own query string to the URL, continues to work independently of this enhancement.

When the **flash-media-streaming ignore-query-string enable** command is entered, for every request in which the query string has been ignored, a message is written to the FMS error log, and the Query String Bypassed counter is incremented in the output of the **show statistics flash-media-streaming** command. The FMS access log on the edge SE contains the original URL before the query string was removed.

The **flash-media-streaming ignore-query-string enable** command affects every VOD and live streaming request and is not applicable to proxy-style requests.

Interactive Applications

Flash Media Streaming supports pass-through (proxy) functionality for interactive applications (non-VOD and non-live). The interactive applications are hosted on a Flash Media Interactive Server that is external to the VDS-IS.

Note For the edge server proxy to function correctly, the origin server must be running Adobe Flash Media Server 3.5.

Direct routing from the Service Engine, acting as the Flash Media Streaming edge server proxy, to the origin server (the Flash Media Interactive Server) is supported by way of the hierarchical path of Service Engines to the origin server. Every Service Engine that receives the request proxies it to the next SE along the path, where it reaches the origin server. Using the Delivery Service framework, the origin server is abstracted from the client request by using the Service Router Domain Name (SRDN), which resolves to the Service Engine that accepts the user connection and forwards the request to the origin server. Flash Media Streaming includes the edge server (proxy) mode, and by default, all non-live and non-VOD applications are proxied by using the edge server. Flash Media Streaming selectively picks connections for processing in edge server mode and aggregates connections to the origin servers.

Note The video and audio content used in an interactive application is cached on the SE acting as the Flash Media Streaming edge server proxy and is not removed when Flash Media Streaming is disabled. The maximum storage allowed for cached content associated with interactive applications is 2 GB. The only way to delete this cached content is to use the **clear cache flash-media-streaming** command or to reload the VDS-IS software on the SE.

VDS-IS supports implicit URI as the method that allows the client to connect with the edge server without exposing the origin server. The URI would look like this: `rtmp://edge1.fms.com/ondemand`.

Request routing based on SWF files or using RTMP redirection is supported. However, RTMP redirection requires more changes in the client code. SWF file-based redirection is recommended. SWF redirection works as follows:

1. The SWF files and associated HTML pages are either prefetched or hosted in the origin server.
2. The client uses a web browser to access the HTML page, which also loads the SWF file.
3. The SWF file is accessed using the SRDN.
4. The Service Router redirects the request to a Service Engine.
5. The SWF file is downloaded to the web browser.
6. The ActionScript in the SWF file attempts to connect to the same host from where the SWF file was downloaded. This is an RTMP connection that reaches the Service Engine.
7. The Service Engine checks for the application type in the URI, and if it is not VOD or live, the processing is moved to the edge server mode and the connection is forwarded to the origin server.
8. The Service Engine tunnels the data between the client and the origin server.

Note Changes to a Delivery Service do not affect existing connections to the Flash Media Interactive Server (origin server). Only new connections are affected by changes to a Delivery Service.

Note URL signing for interactive applications is supported. For more information, see the "URL Signing and Flash Media Streaming" section on page I-13.

Service Router

The Service Router has three parts:

- Request Routing Engine, page 1-34
- Proximity Engine, page 1-47

The Service Router can be configured as both the Request Routing Engine and the Proximity Engine, or the Service Router can be configured only as the Request Routing Engine. Additionally, the Service Router can act as a standalone Proximity Engine by not configuring the Request Routing Engine as the authoritative DNS server.

The Proximity Engine contains the functionality of the Proximity Servers used for proximity-based routing. See the "Proximity-Based Routing" section on page 1-41 for more information on this routing method. The Proximity Engine peers with network routers and listens in on route updates to get topology and routing path information. This information is used to locate the closest resource in the network. Real-time measurements of reachability and delay are also considered. See the "Proximity Engine" section on page 1-47 for more information on the Proximity Engine.

Request Routing Engine

The *Request Routing Engine* mediates requests from the client devices and redirects the requests to the most appropriate Service Engine. It monitors the load of the devices and does automatic load balancing.

The Request Routing Engine is the authoritative Domain Name System (DNS) server for the routed request for the fully qualified domain name (FQDN) of the origin server. In other words, the Request Routing Engine responds to any DNS queries for that domain.

Routing Redirection

There are three ways for client requests to get routed to the Request Routing Engine and on to the Service Engine:

- Router fully qualified domain name (RFQDN) redirection
- DNS-based redirection
- IP-based redirection

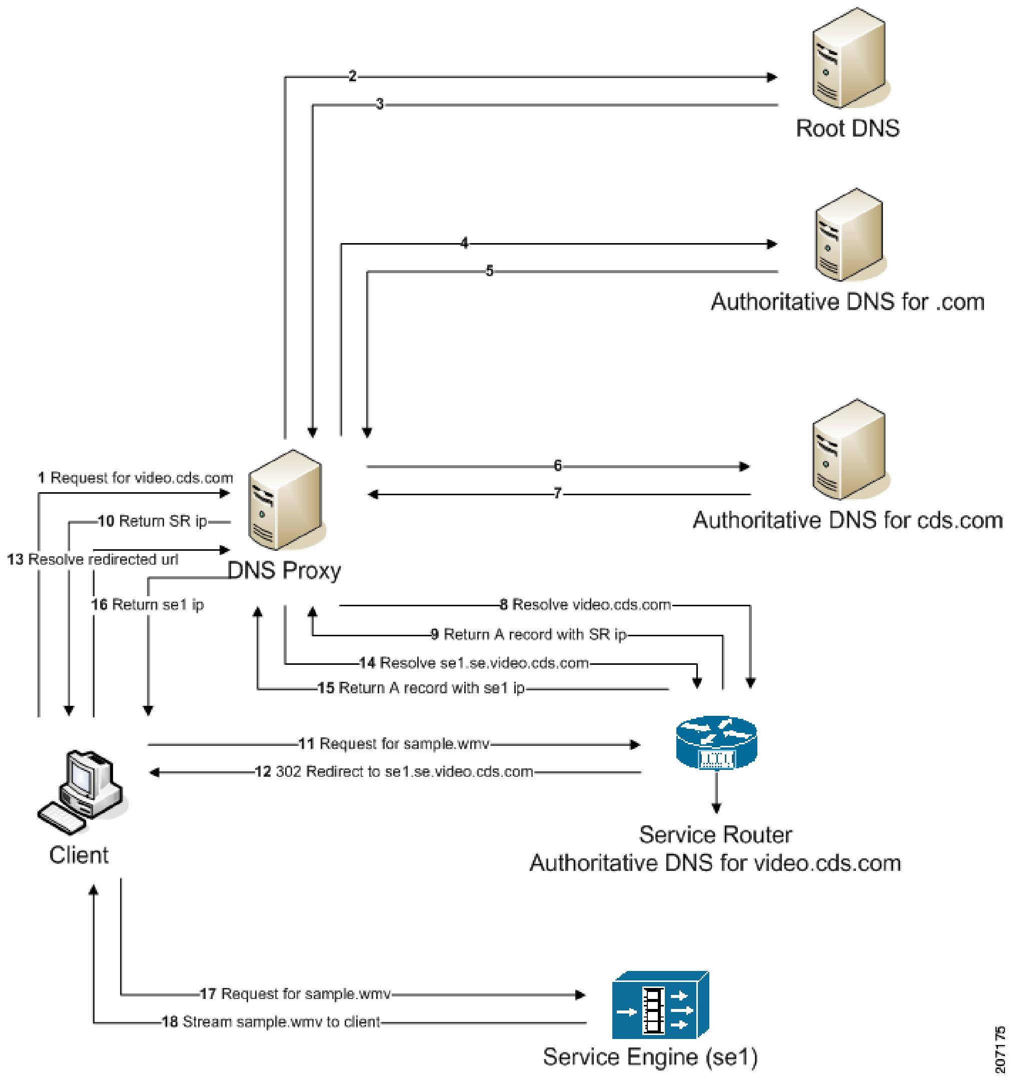
RFQDN Redirection

RFQDN redirection is the default configuration. With RFQDN redirection, client requests are resolved to the Request Routing Engine by the DNS server and the Request Routing Engine redirects the request to the Service Engine based on route tables created from the Coverage Zone file and the current load of the Service Engines. The redirected URL is http://SENAME.SE.RFQDN/relative_path_of_content, where SENAME is the hostname of the Service Engine.

Note The redirected URL for Flash Media Streaming requests is: rtmp://SENAME.SE.RFQDN/application_name/encoded (relative_path_of_streamname), where SENAME is the hostname of the Service Engine.

Figure 1-2 describes the Request Routing Engine workflow for RFQDN redirection.

Figure 1-2 Request Routing Engine Workflow for RFQDN Redirection


2
3
Root DNS
4
5
Authoritative DNS for .com
1 Request for video.cds.com
10 Return SR ip
13 Resolve redirected url
DNS Proxy
16 Return se1 ip
6
7
Authoritative DNS for cds.com
8 Resolve video.cds.com
9 Return A record with SR ip
14 Resolve se1.se.video.cds.com
15 Return A record with se1 ip
11 Request for sample.wmv
12 302 Redirect to se1.se.video.cds.com
Client
Service Router
Authoritative DNS for video.cds.com
17 Request for sample.wmv
18 Stream sample.wmv to client
Service Engine (se1)
207175

In Figure 1-2, the client sends a request for a video file (for example, sample.wmv) to http://video.cds.com. The browser in turn sends a recursive DNS request to resolve video.cds.com through the DNS proxy.

The Service Router is configured to be the authoritative DNS for video.cds.com. The DNS proxy resolves video.cds.com to the Service Router's Request Routing Engine and sends the Service Router IP address back to the client. The client then sends a request for sample.wmv to the Service Router.

The Request Routing Engine chooses the Service Engine to redirect the request to based on load, location, and other factors. A 302 redirect message is sent to the client with the redirected URL http://se1.se.cds.com/sample.wmv.

A DNS request is sent to the Request Routing Engine again through the DNS proxy to resolve se1.se.cds.com. The Request Routing Engine returns the IP address of se1 to the DNS proxy which is forwarded to the client. The client then contacts the Service Engine (se1) directly and requests the sample.wmv. The Service Engine streams the requested content to the client.

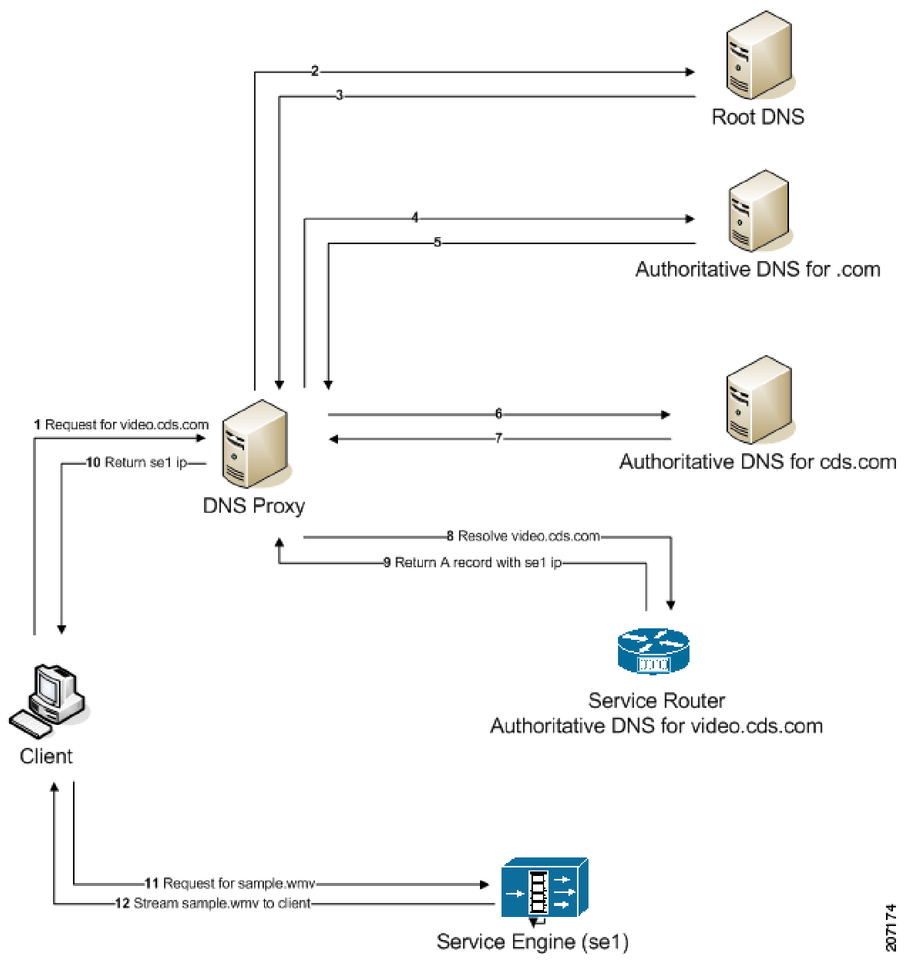
DNS-Based Redirection

DNS-based redirection enables requests to get directly routed to the Service Engine without any 302 redirects. It also allows content to be streamed without transforming the request URL.

Note When DNS-based redirection is used, for application-level requests, last-resort redirection is supported. However, on the DNS plane, an A record with the last-resort domain name or IP address is not returned.

Figure 1-3 describes the Service Router's Request Routing Engine workflow using DNS-based redirection.

Figure 1-3 Request Routing Engine Workflow with DNS-Based Redirection


2
3
Root DNS
4
5
Authoritative DNS for .com
1 Request for video.cds.com
10 Return se1 ip
DNS Proxy
6
7
Authoritative DNS for cds.com
8 Resolve video.cds.com
9 Return A record with se1 ip
Service Router
Authoritative DNS for video.cds.com
Client
11 Request for sample.wmv
12 Stream sample.wmv to client
Service Engine (se1)
207174

When DNS-based redirection is enabled, the DNS proxy contacts the Request Routing Engine to resolve video.cds.com (step 8 in Figure 1-3), the Request Routing Engine determines which Service Engine to redirect the request to based on load, location, and other heuristics, and directly returns the appropriate Service Engine's IP address instead of the Service Router's IP address. The client then directly requests the content from the Service Engine instead of the Service Router.

Note The TTL for the DNS proxy requests is one second. A one-second TTL ensures that the DNS proxy keeps sending requests to the Request Routing Engine, which in turn causes the Request Routing Engine to determine the best Service Engine at that point in time, and not to redirect the request to the same SE.

Note There are certain side effects in adopting this approach. They are as follows:

- When creating the Coverage Zone file, the IP address of the DNS proxy needs to be used for the client IP address range.
- If proximity-based routing is enabled, it uses the IP address of the DNS proxy in computing the proximity.
- If location-based routing is enabled, the location of the DNS proxy is taken into consideration in the selection of the SE.
- Service-aware routing cannot be used because the protocol and content type are not considered at the DNS level.
- Content-based routing cannot be used because the protocol and content type are not considered at the DNS level.

To configure DNS-based redirection, use the **service-router redirect-mode dns-redirect** command.

service-router redirect-mode dns-redirect {**all** | **domain** *domain*}

The following example enables DNS-based redirection with the cdsfms.com domain as the domain used to redirect all client requests to:

```
SR(config)# service-router redirect-mode dns-redirect domain cdsfms.com
```

To display information about the redirect mode, use the **show service-router redirect-mode** command.

To display the statistics, use the **show statistics service-router summary** command and the **show statistics se** command. The output for the DNS-Based Redirection feature is listed as DNS Requests. In addition to these two show commands, there is also the **show statistics service-router dns** command.

IP-Based Redirection

When IP-based redirection is enabled, the Request Routing Engine uses the IP address of the Service Engine in the URL instead of the hostname. The redirected URL is http://<se ip addr>/ipfwd/<rfqdn>/<path>. The IP-based redirection method avoids the extra DNS lookup that was required in the RFQDN redirection.

Note The Web Engine does not support IP-based redirection.

Off-Net and On-Net Clients

The Request Routing Engine chooses the Service Engine based on two scenarios:

- Client is directly connected to the service provider’s network (on-net).
- Client is roaming outside the home network (off-net).

When clients are connected to the service provider’s network, the Service Engine is chosen based on the requested FQDN, the client’s IP address, and the responsiveness of the Service Engine. The Request Routing Engine compares the client’s IP address against a table of address ranges representing the client subnets assigned to each Service Engine. This table is known as the *Coverage Zone file*. The Coverage Zone file provides information on the proximity of the client to the Service Engine based on each client's IP address.

If the client is not connected to the service provider network and location-based routing is enabled, the Request Routing Engine compares the latitude and longitude of each Service Engine, which is defined in the Coverage Zone file, with the latitude and longitude of the client, which is obtained from the Geo-Location servers, to assign a Service Engine that is geographically closest to the client. For more information, see the "Location-Based Routing" section on page 1-41.

Coverage Zone File

When a Service Engine is registered to the CDSM, it is assigned a default Coverage Zone file that is created by the CDSM using the interface IP address of the Service Engine. The default Coverage Zone file can be unassigned, and a custom coverage zone can be created using the Coverage Zone file.

A Coverage Zone file is an XML file containing coverage zone entries for each client IP address range, the Service Engine serving that range, the latitude and longitude of the Service Engine, and a metric value. The Coverage Zone file can be referenced by a URL and imported into the CDSM, or uploaded from a local machine. The Coverage Zone file can be set as the default for a specific Service Router or for all Service Routers in the VDS-IS network.

When content is requested by a client, the Request Routing Engine checks the client's IP address to find the coverage zone that contains that IP address. The Request Routing Engine then selects the Service Engine that serves this coverage zone.

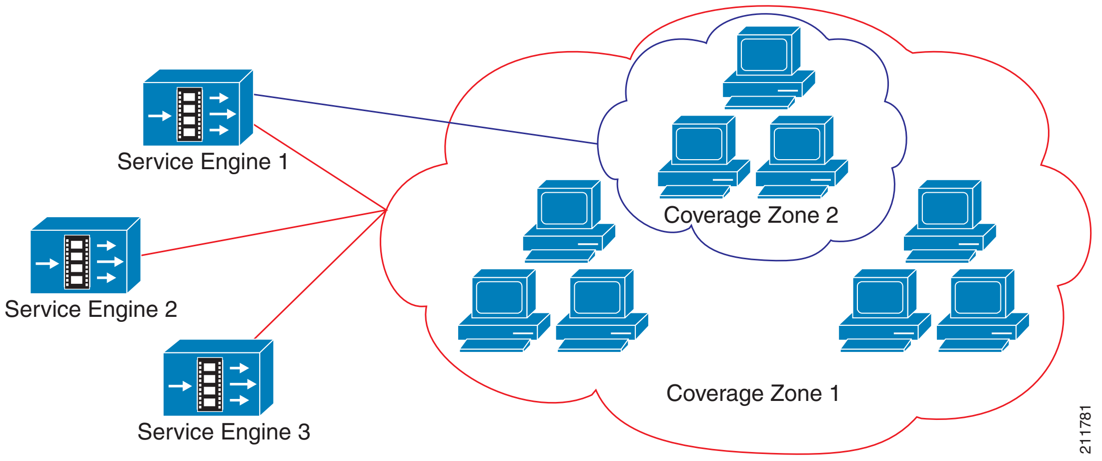
Note When DNS-based redirection is enabled, the Coverage Zone file needs to have entries with respect to the IP address of the DNS proxies instead of the client IP address.

If a specific IP address is in multiple coverage zones, the one with the more specific range is selected. If no match is found in the coverage zone data and if location-based routing or proximity-based routing is enabled on the Request Routing Engine, the Request Routing Engine looks up the best Service Engine closest to the client. If the Request Routing Engine is unable to redirect the request, the Request Routing Engine sends an error response to the client.

A coverage zone can be associated with one or more Service Engines. Each Service Engine can have its own unique coverage zone, or the Service Engines can be associated with more than one coverage zone and have over lapping coverage zones.

In Figure 1-4, all Service Engines serve Coverage Zone 1, and Service Engine 1 is specifically associated with Coverage Zone 2, a subset of Coverage Zone 1.

Figure 1-4 Coverage Zone Example



If a coverage zone is served by multiple Service Engines, all Service Engines are put in the routing table. The metric value, entered in the Coverage Zone file, indicates the proximity of the Service Engine to the client. When multiple Service Engines serving a coverage zone are on the same subnet and have the same metric value, and load-based routing is not enabled, the Request Routing Engine uses round-robin routing to redirect the client. If load-based routing is enabled, the load of the Service Engines are used to determine the best Service Engine to redirect the client.

Routing Methods

The Request Routing Engine chooses the best Service Engine based on whether the Service Engine is participating in the Delivery Service for which the origin server matches that of the requested domain, and whether the Service Engine is assigned to serve the client's network region, as defined in the Coverage Zone file.

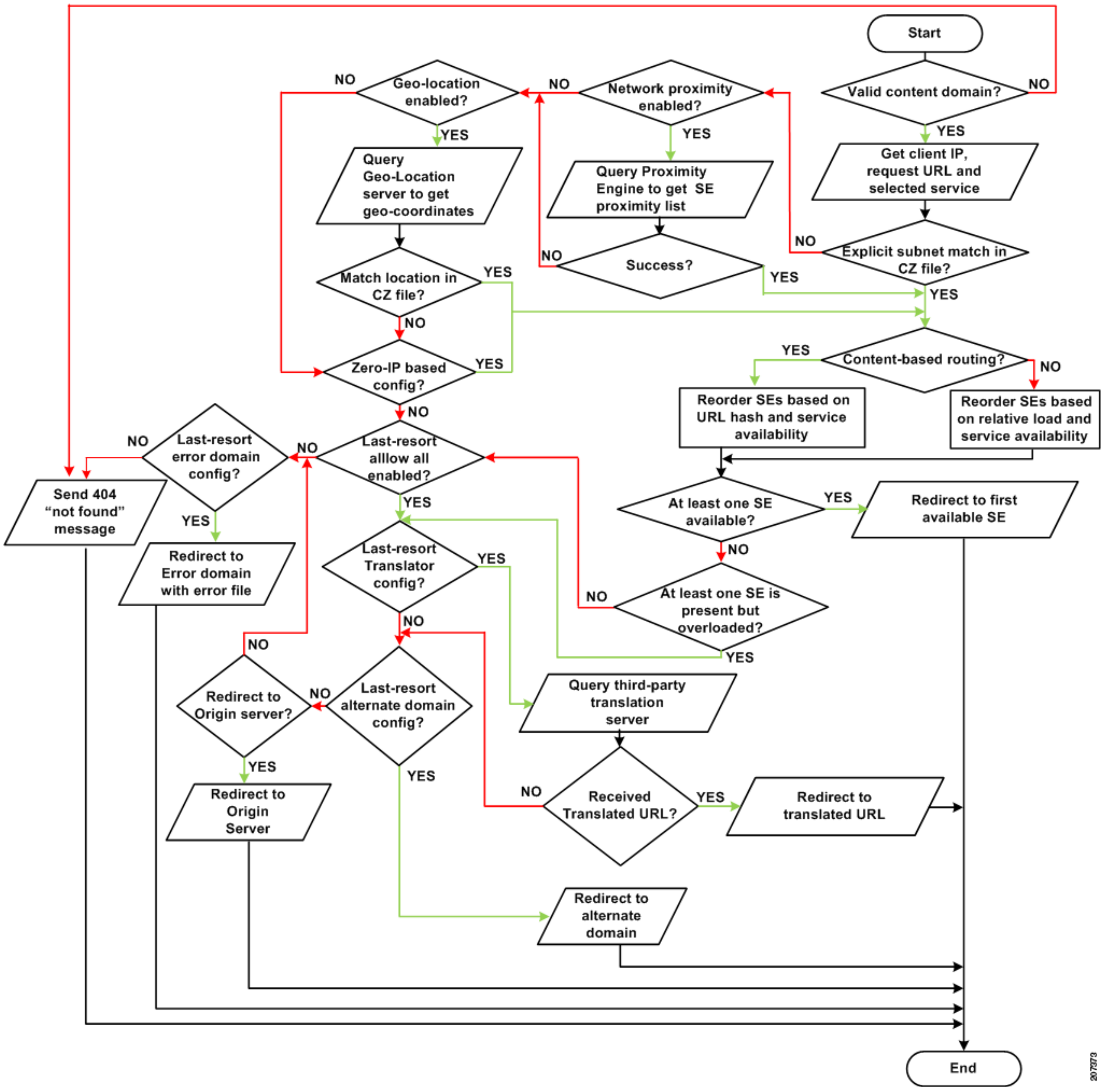
If the client's subnet is not defined in the Coverage Zone file, the Request Routing Engine checks the following routing methods to see if they are configured:

- Load-Based Routing, page 1-40
- Proximity-Based Routing, page 1-41
- Location-Based Routing, page 1-41
- Zero-IP Based Configuration, page 1-41
- Last-Resort Routing, page 1-42
- Service Aware Routing, page 1-43
- Content-Based Routing, page 1-44

Note The keepalive messages between the Service Router and Service Engine are transmitted and received on port 2323. However, the software inter-operates with older software releases that do not use port 2323 for keepalive messages. If a firewall is configured between the Service Engine and the Service Router, port 2323 (UDP) must be opened for the keepalive message to go through.

Figure 1-5 describes the order in which the different routing methods are addressed in the Request Routing Engine.

Figure 1-5 Request Routing Engine Workflow of Routing Methods


Start
Valid content domain?
NO
YES
Get client IP, request URL and selected service
Explicit subnet match in CZ file?
NO
YES
Network proximity enabled?
NO
YES
Query Proximity Engine to get SE proximity list
Success?
NO
YES
Geo-location enabled?
NO
YES
Query Geo-Location server to get geo-coordinates
Match location in CZ file?
YES
NO
Zero-IP based config?
YES
NO
Content-based routing?
YES
NO
Reorder SEs based on URL hash and service availability
Reorder SEs based on relative load and service availability
At least one SE available?
YES
NO
Redirect to first available SE
At least one SE is present but overloaded?
NO
YES
Last-resort alllow all enabled?
NO
YES
Last-resort error domain config?
NO
YES
Send 404 "not found" message
Redirect to Error domain with error file
Last-resort Translator config?
YES
NO
Query third-party translation server
Received Translated URL?
NO
YES
Redirect to translated URL
Last-resort alternate domain config?
NO
YES
Redirect to alternate domain
Redirect to Origin server?
NO
YES
Redirect to Origin Server
End
207373

Load-Based Routing

Load-based routing is enabled by default and cannot be disabled. In load-based routing, the routing decision is made according to the capacity and load of the Service Engines.

The load of the Service Engine is determined by different parameters, such as processor usage, memory usage, disk usage, the number of current Windows Media streams being served, and so on. The current load is compared with the thresholds configured for the Service Engine. If a threshold has been exceeded for a Service Engine it is excluded from the routing table.

Note Bursts of traffic (such as 800 connections per second) may cause the Web Engine to become disabled before it can transmit notification to the SR that the threshold has been reached.

Proximity-Based Routing

Proximity-based routing offers more intelligence to service routing by using network proximity for Service Engine selection. In proximity-based routing, the Request Routing Engine uses the collocated Proximity Engine, or an external Proximity Server that runs routing protocols to get route updates from network routers. A Proximity Server listens for OSPF, BGP, and IS-IS updates and provides proximity information between clients requesting content and Service Engines that have the requested content. It provides a list of Service Engines to the Request Routing Engine ranked in order of optimal routes for content and messages in a network.

Proximity-based routing is used to select the closest Service Engine for a specified client IP address. The Proximity Engine and Proximity Server communicate with network routers and listen in on route updates and gets topology and routing path information. This information is used to locate the closest resource in the network. Real-time measurements of reachability and delay are also considered.

For information on the collocated Proximity Engine, see the "Proximity Engine" section on page 1-47.

Location-Based Routing

Location-based routing is used for off-net clients. Off-net clients are clients that are not directly connected to the service provider network. Location-based routing is designed to work with load-based routing. When both are enabled, the Request Routing Engine first looks up the client IP address in the Coverage Zone file. If there is no subnet match, the client's geographical location is compared to the geographical location of the Service Engines listed in the Coverage Zone file, and the closest and least-loaded Service Engine is selected. Geographically locating a client is used when users roam outside of their home network.

To provide routing to off-net clients, the Request Routing Engine communicates with a Geo-Location server, which maps IP addresses to a geographic location. For redundancy, the CDSM can be configured with a primary and secondary Geo-Location server.

The Geo-Location server identifies the geographical location of an off-net client by the latitude and longitude of the client. The Request Routing Engine compares the client's location with the location of the Service Engines participating in that Delivery Service and chooses the best Service Engine to serve the content.

Zero-IP Based Configuration

The zero-ip based configuration is a catch-all condition for routing. It can be used in combination with proximity-based routing and location-based routing. If an SE cannot be found through location-based routing or proximity-based routing, the zero-ip based configuration is taken into account for selecting an SE.

The zero-ip based configuration is a network entry in the Coverage Zone file defined as 0.0.0.0/0. It matches all client subnets. If the client subnet does not match any of the other network entries in the Coverage Zone file and a 0.0.0.0/0 network entry exists, then the SEs listed for that entry are considered for serving the client request.

Last-Resort Routing

Last-resort routing is useful when all Service Engines have exceeded their thresholds or all Service Engines in the domain are offline, or the client is unknown. If last-resort routing is configured, the Request Routing Engine redirects requests to a configurable alternate domain or translator response domain when all Service Engines serving a client network region are unavailable, or the client is unknown. A client is considered unknown if the client's IP address is not part of a subnet range listed in the Coverage Zone file, or part of a defined geographical area (for location-based routing) listed in the Coverage Zone file.

Note When DNS-based redirection is used, for application-level requests, last-resort redirection is supported. However, on the DNS plane, an A record with the last-resort domain name or IP address is not returned.

Last-resort routing works dynamically. When the load of one or more Service Engines in the original host domain is reduced below threshold limits or the Service Engines are reactivated, new requests are routed to the original host domain automatically.

Last-resort routing allows redirecting a request to an alternate domain or Origin server (if Enable Origin Server Redirect is enabled) for one of the following conditions:

- All SEs in the Delivery Service have exceeded their thresholds
- All SEs in the Delivery Service are unavailable or no SEs are assigned to the Delivery Service
- The client is unknown

Redirecting to the Origin server is allowed if the Enable Origin Server Redirect field is enabled for the content origin. The default setting is enabled. For more information on this configuration parameter, see the "Content Origins" section on page 5-34.

Note Unknown clients are only redirected to the alternate domain (last-resort domain) or translator response domain when the **Allow Redirect All Client Request** check box is checked or the equivalent **service-router last-resort domain** <*RFQDN*> **allow all** command is entered.

If the last-resort domain or the translator response domain are not configured and the Service Engine thresholds are exceeded, known client requests are redirected to the Origin server (if Enable Origin Server Redirect is enabled) and unknown clients either receive an error URL (if the Error Domain and Error Filename fields are configured), or a 404 "not found" message.

Last-resort routing could also be configured to redirect a client to an error domain and filename.

The URL translator provides a way to dynamically translate the client request URL to redirect the client to a different CDN. With the URL translator option, the following occurs if the SR uses last-resort routing for a client request:

1. The SR contacts the third-party URL translator through the Web Service API. The Web Service API is described in the *Cisco Videoscape Distribution Suite, Internet Streamer 4.2.1 API Guide*.
2. The third-party URL translator sends the translated URL in the response to the SR.
3. The SR sends a 302 redirect message to the client with the translated URL it received from the third-party URL translator.

The timeout for connecting to the URL translator server is 500 milliseconds. There are no retries if the URL translator cannot be reached.

If there is no configuration on the URL translator for the requested domain or the connection timeout threshold has been reached, the SR last-resort routing falls back to the alternate domain configuration.

Alternate domain last-resort routing supports requests from RTSP, HTTP (including MMS-over-HTTP), and RTMP clients.

URL translator last-resort routing supports RTSP and HTTP client requests. For Flash Media Streaming clients (RTMP), the client must be able to handle redirects to a different application name. Most Flash clients cannot support a stream name change; so the filename returned by the translator is ignored.

Service Aware Routing

Service-aware routing is enabled by default and cannot be disabled. In service aware routing, the Request Routing Engine redirects the request to the Service Engine that has the required protocol engine enabled, the required protocol engine is functioning properly and has not exceeded its threshold, and the SE has not exceeded its thresholds as configured. See the "Setting Service Monitor Thresholds" section on page 4-89 for more information.

The following user agents are served by the Windows Media Engine:

- Natural Selection (NS) player and server
- Windows Media player and server

The following user agents are served by the Movie Streamer Engine:

- QuickTime player and server
- RealOne player
- RealMedia player

Note In addition to redirecting requests based on the user agents listed in this section, requests for content with the following file extensions are served by Windows Media Engine (both HTTP and RTSP requests):

- wma
- wmv
- asf
- asx

Requests for content with the following file extensions are served by the Movie Streamer Engine:

- 3gp
- 3gp2
- mov
- mp4

When a request reaches the Service Router, the Request Routing Engine generates a hash from the URI. The Request Routing Engine first generates a list of Service Engines to best serve the request based on service aware routing. The Request Routing Engine then reorders the list based on the hash and selects the best Service Engine. Because the hash generated for the same URI is equal, typically the same Service Engine is selected. If the Service Engine is overloaded, the next Service Engine in the list is selected.

For service aware routing, some of the services running on a Service Engine are protocol based. When protocol-based services associated with a protocol engine are stopped on a Service Engine, the Request Routing Engine excludes this Service Engine from the list of possible Service Engines that can serve requests for this type of content. The Request Routing Engine identifies the protocol engine that serves the request based on the user-agent in the request. For example, if some Windows Media Engine-related services are stopped, the Service Engine can still serve Web Engine requests. However, if the request for Web Engine content is sent from a Windows Media Player, the Request Routing Engine excludes the Service Engine from the list of possible Service Engines that can serve the request.

Note If the Web Engine is disabled on the Service Engine, the Service Engine does not get selected for serving any requests, including Windows Media Streaming, Flash Media Streaming, and Movie Streamer.

Note For service aware routing, if a threshold is exceeded for all Service Engines, the Request Routing Engine redirects the client request to the origin server if a last-resort alternate domain is not configured. If a last-resort alternate domain is configured, the alternate domain takes precedence over the origin server. For a managed-live URL, if the origin server does not match the source of the live program, the above case fails. For the above case to work, the origin server host must be configured to match the live program source. In addition, the origin server stream name must be the same as the live program name.

Content-Based Routing

In content-based routing, the Request Routing Engine redirects the request based on the URI. Requests for the same URI are redirected to the same Service Engine, provided the Service Engine's thresholds are not exceeded. If the same SE is not available, requests are routed to the next best SE. If the original SE is available again, requests are routed back to it irrespective of the number of interim redirects to the second best SE.

The same content can be stored in more than one Service Engine if the number of redundant copies is set to more than one. Redundancy is used to maximize the cache-hit ratio. If redundancy is configured with more than one copy, multiple Service Engines are picked for a request with the same URI hash.

Content-based routing is best suited for cache, prefetched, and live program requests to maximize the cache-hit ratio.

Note A client RTMP URL request for Flash Media Streaming does not contain the stream name; therefore, a client's URL requests for different RTMP streams could seem the same. For this reason, content-based routing may not be efficient for Flash Media Streaming because a different directory needs to be created for each stream to differentiate the content.

Note Content-based routing does not work with clients sending signed URL requests. The hashing algorithm for content-based routing considers the whole signed URL, so a signed URL request for the same content may be redirected to a different SE.

Request Routing Engine Workflow of Coverage Zone, Proximity-Based Routing, and Location-Based Routing

The Request Routing Engine workflow for clients connected to the service provider's network is as follows:

1. The client sends the DNS query for the routed FQDN to the local DNS server.
2. The DNS server replies with the Service Router IP address.
3. The client issues an HTTP, RTMP, or RTSP request to the Service Router.

4. If the Request Routing Engine finds the client's subnet in the Coverage Zone file, the following occurs:
   a. The Request Routing Engine chooses the appropriate Service Engine and performs a protocol-specific redirection.
   b. The client issues an HTTP, RTMP, or RTSP request to the Service Engine.
   c. The Service Engine serves the content.

   If the Request Routing Engine does not find the client's subnet in the Coverage Zone file and proximity-based routing has been enabled, the following occurs:
   a. The Request Routing Engine communicates with the Proximity Engine and gets the SE proximity list with the SEs that have the least network cost listed first.
   b. The Request Routing Engine selects the closest Service Engine for the specified client IP address.
   c. The Request Routing Engine performs a protocol-specific redirection with the closest Service Engine.
   d. The client issues an HTTP, RTMP, or RTSP request to the Service Engine.
   e. The Service Engine serves the content.

   If the Request Routing Engine does not find the client's subnet in the Coverage Zone file and location-based routing has been enabled, the following occurs:
   a. The Request Routing Engine communicates with a Geo-Location server and gets the geographical coordinates of the client's IP address.
   b. The distance is calculated between the client and the Service Engines, and the Service Engine closest to the client is selected.
   c. The Request Routing Engine performs a protocol-specific redirection with the closest Service Engine.
   d. The client issues an HTTP, RTMP, or RTSP request to the Service Engine.
   e. The Service Engine serves the content.

When a Service Router is registered with the CDSM, the CDSM propagates the Service Router's IP address to all of the registered devices. The Service Engine sends a keepalive message to the Service Router on a periodic interval, which consists of information about the SE resources (such as disk, CPU, memory, and network interface usage). The Request Routing Engine uses the Service Engine's load and liveness information for generating the routes.

The VDS-IS can have more than one Service Router to support Service Router failover. In line with failover, the DNS server should be configured with multiple Service Routers for the same routed FQDN.

Note DNS entries for all FQDNs must be delegated to the Service Router. In the DNS server's database file, a name server record must be entered for each FQDN that routes to the Service Router.

Request Redirection

The Request Routing Engine supports the following redirections:

• HTTP ASX Redirection	Used if the requested file has an.asx extension. This redirection method is used for Windows Media Technology. To use the HTTP 302 redirection instead, see the "Configuring Application Control" section on page 4-128.
• HTTP 302 Redirection	Used if the protocol is HTTP and the file extension is not .asx. This is the native HTTP redirection.
• RTSP 302 Redirection	Used if the protocol is RTSP and the client is QuickTime or Windows Media. This is the native RTSP redirection.
• RTMP 302 Redirection	Used if the protocol is RTMP and the client is Adobe Flash Player, Adobe Media Player, or Adobe Flash Lite Player.

Normal requests for files with an .asx extension returns a status 200, unless HTTP 302 redirection is enabled.

Cross-Domain Policy

For Flash Media Streaming, when a client requests content from a portal. and the content contains a request to a different remote domain (the origin server in the case of the VDS-IS), the request cannot be served unless the remote domain (origin server) has a crossdomain.xml that grants access to the original portal.

For example, if a client request is for abc.com/streaming.html, and the content in streaming.html has a request to cds-origin.com/vod/sample.flv, the client requests a crossdomain.xml. The crossdomain.xml allows access to abc.com, which allows the streaming of sample.flv.

If the cds-origin.com does not have crossdomain.xml, then the request is denied.

Note For Flash Media Streaming, the remote domain request is looked up in the crossdomain.xml file. For Microsoft Silverlight, the remote domain request is looked up in the clientaccesspolicy.xml file.

In the VDS-IS, instead of directly going to cds-origin.com, the request first comes to the Service Router. When the request for crossdomain.xml comes to the Service Router, the Request Routing Engine sends it to the client. This XML file grants access to the portal for the file requested. The client then sends the request for the file, which is then served.

Note For Windows Media Streaming Silverlight the clientaccesspolicy.xml file is requested only when web service calls are made. Depending on the client player, for both Windows Media Streaming Silverlight and Flash Media Streaming applications, the clientaccesspolicy.xml and crossdomain.xml need to be provisioned on the origin server.

Note Flash Media client players that use FLVPlaybackComponent do not currently request the crossdomain XML file for video files. The crossdomain request is issued only when a query string is present. In such cases, the video gets downloaded but does not play.

Configuring and Monitoring the Cross-Domain Policy Feature

The Cross-Domain Policy feature can be enabled through the CDSM. See the "Configuring Cross-Domain Policy" section on page 4-116 for more information.

Logging information can be found in the /local/local1/errorlog/service_router_errorlog.current file. When the Request Routing Engine sends the crossdomain.xml to a client, the "crossdomain.xml served to client" message is logged. When the Request Routing Engine sends the clientaccesspolicy.xml file to a client, the "clientaccesspolicy.xml served to client" message is logged.

The **show statistics service-router summary** command displays an increase in the number of the HTTP Requests (normal) in Request Received section of the output.

Note The crossdomain.xml or clientaccesspolicy.xml file served by the SR is logged as 200 OK, and the request redirect is logged as a 302.

Unified Routing Table

The unified routing table uses one global route context for all domains, with all SEs from the Coverage Zone file added to one set of route tables. This is a good option if the VDS-IS serves a number of different domains (configured as delivery services), but uses the same set of SEs for the different delivery services (domains). By enabling the unified routing table in this scenario, the memory usage on the SR is reduced.

If the VDS-IS is configured with fewer domains or the SEs are not all serving the same domains, then the memory usage is not impacted as much, and not enabling unified routing may be a better option.

The unified routing table option is disabled by default. To enable unified routing on the SR, enter the **service-router unified-routing -table enable** command.

Note Not enabling the unified routing table increases the memory usage on the SR. Make sure the memory usage does not exceed the recommended limit, which is 1.5 GB when the SR is running with a load and no configuration changes are occurring.

Proximity Engine

The Proximity Engine leverages routing information databases (IGP and BGP) by interconnecting and peering with routers. This information is used to compute the network distance between a source address, referred to as the proximity source address (PSA) and a target address, referred to as the proximity target address (PTA). This distance is known as the *proximity rating of the PTA*.

The Proximity Engine is configured as one of the Proximity Servers for the proximity-based routing method. See the "Proximity-Based Routing" section on page 1-41 for more information.

Note The Proximity Engine only participates in the Open Shortest Path First (OSPF), Intermediate System-to-Intermediate System (IS-IS), and Border Gateway Protocol (BGP) to gather information to make proximity decisions. The Proximity Engine is not a router and does not ever use the information to route traffic.

Note The Proximity Engine is only supported on the CDE205 and the CDE220-2G2 platforms.

The standby interface is not supported for Proximity Engine. Use port channel configuration instead.

The Proximity Engine operates in an IP routing domain where the Interior Gateway Protocol (IGP) or BGP is used to distribute routing information across the domain. For the proximity function to work, at least one of the following is required:

- Enabled link-state protocol, such as OSPF or IS-IS for IGP proximity, which is required if the Proximity Engine is going to peer with IGP routers.
- Enabled policy routing protocol, such as BGP for best-path proximity and location-community proximity, which is required if the Proximity Engine is going to peer with BGP routers.

Note All BGP routes must resolve to IGP next hops or directly connected routes.

Routers running OSPF or IS-IS establish adjacencies with their directly connected neighbors and exchange their connectivity view (that is, each router advertises its visibility about its adjacencies). Advertisements are flooded throughout the whole routing area and each router stores each received advertisement in a link-state database (LSDB).

The LSDB contains the topology of the whole network and each router uses it to compute the Shortest Path Tree and the Routing Information Base (RIB) that contains each known IP prefix in the network and its corresponding next-hop.

OSPF and IS-IS are the two IP link-state protocols. They operate quite similarly:

- Establish adjacencies with directly connected neighbors
- Create a routing update packet (OSPF LSA and ISIS LSP) containing the router connectivity
- Flood routing updates (LSA or LSP) throughout the routing area
- Collect all received routing updates in a LSDB
- Compute shortest first path (SPF) algorithm
- Populate the RIB with the result of SPF

The difference between OSPF and IS-IS is in the way packets are formatted and exchanged (encapsulation) and in the way adjacencies and flooding are handled over broadcast media. From a routing computation perspective, both protocols behave the same and therefore the Proximity Engine operates the same in networks deploying OSPF or ISIS.

The Proximity Engine makes proximity decisions using information from the same link-state database that is passed between routers using OSPF or IS-IS. For these reasons, the Proximity Engine must be configured to make either OSPF or IS-IS adjacencies to gather link-state database information for routers in the same autonomous system, and BGP adjacencies to gather the BGP routing information for routers in the different autonomous systems.

Proximity Engine Request Flow

Following is the Proximity Engine request flow:

1. The Request Routing Engine sends the proximity request to the Proximity Servers, the first of which could be the collocated Proximity Engine.

The proximity request specifies a PSA (the client's IP address) and a set of one or more PTAs (IP addresses of the SEs).

2. The Proximity Engine receives the proximity request and performs a route lookup on the PSA.
3. The Proximity Engine determines whether the request should be handled by IGP, BGP, or locally. Local routing is used when both the PSA and PTA are both local to the network. If the proximity algorithm for BGP location community is enabled, and the PSA has community attribute information, then both BGP and IGP routing information is considered.

   The Proximity function takes into account:
   - Routing topology
   - Inter-autonomous system reachability
   - Optimal path taken by the requested data
4. The Proximity Engine sends the response back to the Request Routing Engine.

   In the proximity response, the Proximity Engine returns a list of proximity target addresses and the cost associated with each address. This list includes all of the IP addresses of all of the SEs registered to the CDSM. Using the proximity response data, the Request Routing Engine can select the closest (best) target.

Note If multi-port support is configured on the with multiple IP addresses, only one valid IP address of that SE is included in the list. If this is selected, it can load balance the requests among the streaming interfaces.

Proximity Ranking

The proximity ranking could include the following proximity algorithms:

1. BGP community-based proximity
2. BGP best-path proximity
3. IGP proximity

The first two algorithms are only used if they are enabled. The last one, IGP proximity, is enabled when an IGP is configured.

The proximity ranking always contains the proximity target list (PTL) addresses in the same order as above. For example, if there is a PSA and two PTAs (PTA1 and PTA2), and all proximity algorithms are enabled, the following rules are applied:

1. If PSA and PTA1 have at least one community in common and PTA2 does not have a common community, PTA1 is preferred over PTA2.
2. If both PTA1 and PTA2 have at least one community in common as the PSA, the next weight is considered.

   The larger the number, the more weight the community has. If PTA1 has a weight of 5 and PTA2 has a weight of 2, PTA1 is preferred over PTA2.
3. If both PTA1 and PTA2 have the same weight, the next algorithm is considered, which is BGP best-path.
4. For BGP best-path, the PTA with the smallest AS-hop count is preferred. If both PTAs have the same AS-hop count, the next and final algorithm is considered, which is IGP proximity.
5. For IGP proximity, the PTA with the lowest IGP metric is preferred.

BGP Proximity Algorithms

Community-Based Proximity

Two distinct proximity algorithms are used:

- IGP-proximity algorithm gives an ordered list of SE IP addresses known in the IGP database (OSPF or IS-IS).
- BGP-proximity algorithm gives an ordered list of SE IP addresses known in the BGP table.

While the combination of the IGP and BGP basic proximity is sufficient for the proximity calculation for most network deployments, they may not be appropriate for some network deployments, such as a Multiprotocol Label Switching (MPLS) network. Most of the time it is sufficient to rank the prefixes and make the recommendation for the prefixes based on whether the PSA and the PTA are in the same rack (the most preferred ranking), the same point of presence (POP), the same city, or the same autonomous system (AS) (the least preferred).

When the BGP community-based proximity option is enabled, additional location information is included in the proximity calculation to influence application-level traffic optimization in the network. When the community-based proximity option is not used, the proximity request is handled by IGP proximity.

The BGP community-based proximity requires that the PSA has a BGP community string. PTAs that have the same BGP community string as the PSA are ranked as more preferred than PTAs that do not have the same BGP community string as the PSA. The association of PSA and PTA community attributes is configurable by specifying the target (PTA) community values to association with the location (PSA) community, and optionally assigning a preference level. For more information, see the "Configuring the BGP Community-based Proximity Settings" section on page 4-124. For the remaining PTAs that have different community strings, they are ranked by IGP proximity.

Best-Path Proximity

Note Best-Path proximity algorithm requires the configuration of the BGP proximity settings.

When the BGP best-path proximity option is enabled, the BGP best-path algorithm ranks each route included in the PTA based on the attribute values associated with each route.

Redirect Proximity

Note Redirect proximity algorithm requires the configuration of the SRP and the BGP proximity settings.

If the PSA is learned from another AS, the current Proximity Engine does not have the best knowledge to handle the request. In this case, if the BGP redirect proximity option is enabled, the Proximity Engine sends back a Redirect response to the Service Router. The Redirect response contains the list of Proximity Engines that reside in the same AS as the PSA. The Service Router then sends the proximity request to one of these Proximity Engines.

Service Routing Protocol

The Service Routing Protocol (SRP) uses distributed hash table (DHT) technology to form a distributed network of Proximity Engines. SRP is highly scalable and resilient. SRP is implemented as an overlay network on top of IPv4 or IPv6 transport. Currently, only IPv4 is supported.

Note SRP is required if the Redirect proximity algorithm is enabled. SRP is used to gather and store information about all of the Proximity Engines that are available for redirection.

A *DHT network* is a logical network composed of Proximity Engines that have the same DHT domain. Although DHT does not play any direct role in responding to the proximity service, it is the integral part of the Proximity Engine system that gathers and stores information about other Proximity Engines in the network to form a cohesive, resilient proximity service network.

Content Delivery System Manager

The Internet Streaming Content Delivery System Manager (CDSM) is a web browser-based user interface. The Internet Streaming CDSM allows the administrator to configure, manage, and monitor delivery services and devices in the Cisco Videoscape Distribution Suite, Internet Streamer (VDS-IS). Application programming interfaces (APIs) are provided for backoffice integration with the Internet Streaming CDSM.

Authentication, Authorization, and Accounting

The Internet Streaming CDSM uses HTTPS to secure the administrator's session. Multiple users can perform administrative operations by using the Internet Streaming CDSM. The administrator can configure certain users to have either view-only rights for monitoring the VDS-IS, or full rights that allow configuration changes as well as monitoring capabilities.

User accounts and groups can be added to the Internet Streaming CDSM and given roles and rights for accessing configuration information. It is also possible to segregate and group objects and give access to a limited group of users.

User authentication can be configured to use RADIUS and TACACS+ servers when available, otherwise the Internet Streaming CDSM provides its own authentication server.

The VDS-IS wide policy and status information is maintained in a relational database on the Internet Streaming CDSM. This information is propagated and synchronized with all devices in the VDS-IS network.

As part of the network management process, the administrator can perform basic administration operations on the Internet Streaming CDSM database, including backup and restore.

Device Management

The Internet Streaming CDSM sends device configuration changes to the selected device or group of devices once the change has been submitted. The device sends any configuration changes that were made locally to the CDSM, and also provides periodic status information.

Devices can be organized into user-defined device groups, which allow administrators to apply configuration changes and perform other group operations on multiple devices simultaneously. Because a device can belong to multiple device groups, this reduces the management overhead of the administrator. Device groups allow for a single instance of management thus eliminating the need to repeat the same step for each device.

The Internet Streaming CDSM also provides an automated workflow to apply software upgrades to the devices in a device group.

Higher Storage Utilization of VDS-IS

Storage across multiple Service Engines is virtually divided into buckets where each Service Engine serves only a subset of the total content. Both the local storage and RAM of the Service Engines can function as an aggregated distributed service, providing unlimited scalability. Linear scaling of the VDS-IS storage is accomplished by adding more Service Engines to one location. This addresses the demands of the "Long Tail" use case relevant to the Service Engines. The Long Tail is the realization that the sum of many small markets is worth as much, if not more, than a few large markets. Long-tail distribution is the possibility that extremely infrequent occurrences in traffic are more likely than anticipated.

This higher storage utilization provides the following:

- Overall better system performance
- Higher in-memory cache hit ratio
- Deterministic resiliency in case of failures or overload due to very popular content (This is useful when customers have live, prefetched, and cached assets more than 4.5 terabytes of content on one Service Engine.)

The content distribution is resilient and stateless. If the load of all content mapped to one Service Engine increases, the load is automatically spread to other Service Engines without requiring any administrator intervention.

Delivery Services Management

The Internet Streaming CDSM provides the configuration and monitoring of delivery services, which defines how content is ingested, stored, cached, and published. The Internet Streaming CDSM provides the Service Engines with information about the delivery services and which Service Engines are participating in the Delivery Service.

In addition to using the Internet Streaming CDSM to define delivery services, an XML file called a *Manifest file* can be used to define a Delivery Service. The Manifest file and APIs serve as the basis for backoffice integration. For more information about the Manifest file, see the "Manifest File" section on page 2-10.

Resiliency and Redundancy

A VDS-IS that is designed with full redundancy and no single point of failure includes redundant Internet Streaming CDSMs and Service Routers. The redundancy mechanisms for the Content Acquirer and Internet Streamer applications running on the Service Engines operate differently.

Content Acquirer Redundancy

In the event of a primary failure on the Content Acquirer, the failover mechanism supports the election of a backup Content Acquirer. A failover requires that both the primary and backup Content Acquirer be located in the root location of the Delivery Service.

Live Programs

If the Content Acquirer receives a live program as a multicast stream from the origin server, upon failure of the primary, the backup Content Acquirer assumes control of that program's streaming and the program continues without interruption. This process is transparent to the end user. When the primary

Content Acquirer comes back online, it receives the live stream from the active secondary Content Acquirer and does not fall back (regain its primary status) until the live program has finished or has been restarted.

If the Content Acquirer receives the program as a unicast stream from the origin server, the failover mechanism is not supported. If the primary Content Acquirer fails while a program is playing, the person viewing the program must re-request the program.

Internet Streamer Redundancy

If a Service Engine running the Internet Streamer application fails, the Service Router stops receiving keepalive messages from that Service Engine. When a new request comes in, the Service Router does not redirect the request to that Service Engine; instead, it redirects the request to other Service Engines within the same Delivery Service. All the existing sessions on the failed Service Engine terminate and the affected end users must re-request the content.

Service Router Redundancy

If the VDS-IS network is designed with multiple Service Routers, all Service Routers are aware of all Service Engines in the VDS-IS. The DNS servers must be configured with multiple Service Routers and the failover is handled by the DNS servers.

Internet Streaming CDSM Redundancy

The Internet Streaming CDSM can operate in two different roles: primary and standby. The primary role is the default. There can only be one primary active in the VDS-IS network; however, you can have any number of Internet Streaming CDSMs operating in standby to provide redundancy and failover capability.

Primary and standby CDSMs must be running the same version of software. We recommend that the standby CDSM be upgraded first, followed by the primary CDSM.

The Internet Streaming CDSM design principle is that the management device is never in the service delivery path. When the CDSM fails, the rest of the VDS-IS continues to operate. A CDSM failure does not affect any services delivered to end users, and all content ingest continues. The only negative effect is that the administrator cannot change configurations or monitor the VDS-IS. As soon as a failure to connect to the CDSM is noticed, the administrator can activate the standby CDSM. For information on making the standby CDSM the primary CDSM, see the "Changing a Standby CDSM to a Primary CDSM" section on page 3-12.