Product Overview

Overview

The Cisco Videoscape Distribution Suite Origin Server (VDS-OS) provides content origination and media processing capabilities on a unified platform that can be managed by the Virtual Origin System Manager (VOSM).

The VDS-OS provides a unified origin solution that can be used to reduce the complexity of heterogeneous Adaptive Bit Rate (ABR) origin servers and traditional content libraries. In addition to supporting some of the major ABR origin functions, it also provides standards-based interfaces for content origination. The VDS-OS system can be your authoritative source of content for different content service ecosystems. It provides a unified ABR content origination framework for on-demand encapsulation (ODE), linear (live) streaming, and time-shifted content. The VDS-OS also has integrated capabilities to convert ABR live video to video-on-demand (VOD) assets based on predefined schedules.

Feature Support

This section lists all of the features in VDS-OS R2.1.1.

- Acquisition, Recording, and Playout of HLS Live and TSTV, page 1-3
  - Load Balance Live Steam Acquisitions from Playout Node to Acquisition Node, page 1-3
  - In-Memory-Live Streaming, page 1-3
  - Stream-Aware Native HLS Pass-Through Mode, page 1-4
  - Multiple Manifest Files for HLS Pass-Through Mode, page 1-5
  - Web Video Text Tracks (WebVTT) Support for HLS Pass-Through Live, page 1-5
  - NDS-HLS Support, page 1-6
- Delivery of HDS, HLS, and HSS VOD Content from NAS, page 1-7
- Delivery of ABR ODE for CF VOD Content on the NAS, page 1-7
  - DRM Support, page 1-7
    - Cisco Key Store 1.1
Chapter 1      Product Overview

Feature Support

- Cisco VMS-KMS
- Verimatrix VCAS 3.3
  - Encryption Enhancements, page 1-7
  - ODE Playout for HDS, HLS, and HSS Formats, page 1-13
    - HTTP-Sourced Common Media Format—cDVR, TSTV (HLS)
    - NAS-Sourced Common Media Format—VOD (HLS)
  - Trick Mode Support for HDS, HLS, and HSS Playback, page 1-13
- Admission Control, page 1-18
- API Management, page 1-19
- Asset Resolution, page 1-19
- Cache-Control Enhancements, page 1-21
  - Cache-Control Headers Support, page 1-21
  - Support for If-Modified-Since for Native ABR for HLS Live and for HDS, HLS, and HSS VOD, page 1-23
  - Support for If-Modified-Since for ODE Playout from CF VOD on the NAS, page 1-25
- External Storage Devices—NAS, page 1-25
- High Availability—Resiliency and Redundancy, page 1-26
- Operational Enhancements, page 1-26
  - Cross-Domain Policy File for HDS Playback, page 1-26
  - Customized Asset URLs, page 1-27
  - Default Asset Resolver File, page 1-20
  - Display Usage for TMPFS and NSS Usage, page 1-29
  - Alarm Enhancements, page 1-29
- Origin Services, page 1-29
- VOSM, page 1-32
  - Authentication, Authorization, and Accounting, page 1-32
  - Device Management, page 1-32
  - Origin Services Management, page 1-33
- Service Engine (SE), page 1-33
  - Media Delivery Applications, page 1-33
  - Linear Stream Acquisition, page 1-33
  - Capture Controller, page 1-33
  - Web Engine, page 1-34
- Service Router (SR), page 1-34
  - Request Routing Engine, page 1-34
  - Coverage Zone File, page 1-38
- Device Modes, page 1-39
- System Topology, page 1-39
Chapter 1  Product Overview

Feature Support

- Virtual Machine (VM) Support, page 1-41
- Provisioning of New Features, page 1-41

Acquisition, Recording, and Playout of HLS Live and TSTV

- Load Balance Live Stream Acquisitions from Playout Node to Acquisition Node, page 1-3
- In-Memory-Live Streaming, page 1-3
- Stream-Aware Native HLS Pass-Through Mode, page 1-4
- Multiple Manifest Files for HLS Pass-Through Mode, page 1-5
- Web Video Text Tracks (WebVTT) Support for HLS Pass-Through Live, page 1-5
- NDS-HLS Support, page 1-6

Load Balance Live Stream Acquisitions from Playout Node to Acquisition Node

The VDS-OS load-balances requests to the Playout node among all active Acquisition nodes, using round-robin routing at a per publish resource level.

The routing occurs between active Acquisition nodes. An Acquisition node is considered active for a publish resource as long as it is able to successfully serve out content for the publish resource. The set of active Acquisition nodes is rotated so that each new client request for which there is no available cached content in the Playout node is routed to a different node, distributing the load evenly across all nodes. Failed nodes are isolated to avoid response time delays, and client requests are retried periodically until successful.

In-Memory-Live Streaming

You can use In-Memory-Live Streaming to optimize or even eliminate the use of NAS. In-Memory-Live Streaming enables you to deploy live services with a small time-shift buffer (DVR window) in RAM. The relatively small size of the buffer means that In-Memory-Live Streaming supports only limited in-session time-shifting.

Services such as start-over based on the time-shift buffer are not supported.

When configuring In-Memory-Live Streaming, keep the following considerations in mind:

- You must configure a live publish resource and a small time-shift buffer on the VOSM’s Publish Resources page.
  - Enter a **Time-Shift Window** size, in minutes, for In-Memory-Live Streaming. The valid range is 1 to 120 minutes. The default setting is 1 minute.
  - Select the **In-Memory Live** checkbox to enable In-Memory-Live Streaming. By default, In-Memory-Live Streaming is disabled.
- You can also use REST APIs on VOSM to configure In-Memory Live publishing with a time-shift buffer.
- For an in-memory publish resource, VDS-OS stores all manifest and segments files in RAM. VDS-OS reserves an appropriate amount of RAM (TMPFS) when the in-memory publish resource is started. If there is insufficient RAM, the in-memory publish resource is not created.
For an in-memory pass-through publish resource, the VDS-OS stores the variant manifests in the NAS, if a NAS is configured for the origin service. Otherwise, the VDS-OS stores the variant manifests on the local disk.

For more information about configuring a publish resource for In-Memory-Live Streaming, see the “Publish Resources” section on page 1-11.

Stream-Aware Native HLS Pass-Through Mode

The VDS-OS supports native HLS Live service in pass-through mode. In pass-through mode, the VDS-OS does not interpret or republish any content received from the HLS encoder or packager or encapsulator (that is, the CMP or CME).

When configuring a pass-through native HLS publish resource, keep the following considerations in mind:

- The VDS-OS publishes the live service using the name that was received from the packager. That is, the Capture Resource URL is same as the Publish Resource URL in pass-through mode.
- The VDS-OS publishes all streams as received from the packager. The VDS-OS does not select or filter streams.
- The time-shift buffer (DVR window) is controlled by the packager. The VDS-OS deletes the segments when it receives an HTTP DELETE from the packager.
- You must configure the time-shift buffer on the VOSM’s Publish Resources page, and it must match the time-shift buffer that is used on the encoder or packager. The time-shift buffer determines when to delete segments, if an HTTP DELETE is not received from the packager, under error conditions.
- You must configure all profile and stream information for a pass-through publish resource on the VOSM’s Publish Resources page.
- When the VDS-OS detects the loss of a stream, as in republish mode, it generates an alarm and updates the status of the capture resource and the channel on the VOSM.
- As a part of HLS Live publishing, the stream manifests are updated as new segments are added to the stream manifest. However, the variant (master) playlist, which describes the stream manifest, is not updated while publishing is in-progress.
- The variant playlist is sent to the VDS-OS only when live publishing starts. Therefore, take special care when the channel is started or restarted on the encoder or packager.
- You must start a live channel or publish resource on the VOSM before you start it on the encoder or packager. This ensures that the VDS-OS resources are ready to ingest content pushed by the encoder or packager.
- If the bit rate changes in HLS Live pass-through mode, you must stop the packager that is running with the old configuration before starting VOSM with the new configuration. When you start the HLS Live channel in VOSM, you must also start the packager within five (5) minutes, or the channel will move to Failed state.
- If a channel or publish resource is stopped on the VOSM, the stopped asset is cleaned up. Therefore, after the publish resource is restarted, the channel must also be restarted on the encoder or packager, to ensure that the variant playlist is published correctly.

For more information about configuring a pass-through native HLS publish resource, see the “Publish Resources” section on page 1-11.
Multiple Manifest Files for HLS Pass-Through Mode

The VDS-OS can support more than one variant playlist for an HLS Live asset. The variant playlists can be used to create different logical assets from the same physical asset, based on devices (such as iPhone or Apple-TV) or policies (such as gold and silver users, or high-definition and standard-definition users).

The VDS-OS allows the logical assets to contain streams that overlap. For example, given profiles A, B, C, and D, one playlist can contain all four profiles; another can contain profiles A, B, and C; and another can contain profiles C and D.

If you have 4 profiles A, B, C and D, then you can have variant manifests where variant1 contains A, B, C and Variant2 contains B, C, D and so on.

To support multiple manifest files, the VDS-OS allows the following VOSM capture resource URLs:

http://vos.domain.net/channel/cap/res/hls/variant.m3u8
http://vos.domain.net/channel/cap/res/hls/

The VDS-OS returns code 2xx for all .m3u8 file ingest requests that fall under a configured capture resource, regardless of whether a pass-through or republish publish resource is configured.

For example, given the above capture resource with two profiles, stream1 and stream2. If the VDS-OS receives ingest requests stream1.m3u8, stream2.m3u8, and variant.m3u8, it returns a 2xx code.

However, the VDS-OS also returns a 2xx code for other .m3u8 ingest requests from a packager. If a corresponding pass-through publish resource is configured for the capture resource, the VDS-OS stores the files in the configured storage, as appropriate. If a pass-through publish resource is not configured, the files are not stored anywhere.

You must configure the multiple variant manifests as multiple presentations.

```xml
<output_stream>
  <stream_id>00000000-0000-0000-0000-00000000a800</stream_id>
  <input_id type="video">1a4d4523-b19a-4c9f-a161-5bdea5990e4d</input_id>
  <input_id type="audio">dda41af9-63a4-4713-833d-834df194af78</input_id>
  <streamname>mid800</streamname>
  <streambitrate>80000</streambitrate>
</output_stream>

<presentations>
  <presentation name="variant1">
    <outputstream stream_id="00000000-0000-0000-0000-00000000a800" />
    <outputstream stream_id="00000000-0000-0000-0000-00000000a1000" />
  </presentation>
  <presentation name="variant2">
    <outputstream stream_id="00000000-0000-0000-0000-00000000a800" />
    <outputstream stream_id="00000000-0000-0000-0000-00000000a1000" />
  </presentation>
</presentations>
```

The expiry or max-age for multiple variants is the same as that for a single variant manifest: the arrival rate.

Web Video Text Tracks (WebVTT) Support for HLS Pass-Through Live

The VDS-OS supports WebVTT for pass-through HLS Live.

WebVTT support requires a .vtt file that is used for subtitles and captioning video content.

- There is one .vtt file per segment per language that is common to all of the profiles.
- Usually, all of the .vtt files for a given language are contained in a folder at the publish resource root, with one folder per language. However, any directory structure works with pass-through.
• There is a separate manifest file that contains all of the .vtt segments per language. The manifest file is usually stored along with the stream manifest files and variant manifest at the publish resource root. The variant manifest contains information about the subtitles that each stream can use.

• The MIME type for .vtt files is text/vtt.

• The .vtt file’s max-age is set to the end of the DVR window, the same as for the .ts file.

• The .vtt manifest file’s max-age is set to the segment arrival rate, the same as for the stream manifest file.

**NDS-HLS Support**

The VDS-OS supports NDS-HLS.

NDS-HLS has a .ptr file, which is a custom manifest file, used by NDS-HLS desktop clients, that contains information similar to that found in stream and variant manifests.

• You can use an ABR-enabled NDS StreamShaper or an ABR-E to POST NDS-HLS with a .ptr file.

• Each live publish resource is associated with its own latest.ptr file that is updated periodically.

• Each VOD asset is associated with an assetname.ptr file.

• We recommend that all of the latest.ptr files be contained in a folder at the publish resource root, the same location where the index.m3u8 file is kept. However, any directory structure works with pass-through.

• The .ptr file’s expiry time is set to the segment arrival rate, the same as for the stream manifest file.

The following sample shows a typical latest.ptr file for a live channel:

```
1.1
Streamshaper/5.8.0.8/196
Live
326E28-326E31 10000 10
0
Mon, 20 May 2013 16:28:36 GMT 531595555200 13295548751
400 400 368 208
600 600 640 368
1200 1200 640 368
2200 2200 720 416
```

The following sample shows a typical .ptr file for a VOD asset:

```
1.1
XTVEncryptor/5.7.0.268/1428
VOD
0-170 10000
0
Mon, 08 Oct 2012 20:39:56 GMT 99497160000 3685080
64 64 320 192
300 300 384 224
600 600 480 272
900 900 640 368
1300 1300 640 480
1600 1600 720 576
```
Delivery of HDS, HLS, and HSS VOD Content from NAS

The VDS-OS accesses VOD content stored on the network-attached storage (NAS) devices managed by the operator. The VOD content is used for recording and DVR buffering.

Delivery of ABR ODE for CF VOD Content on the NAS

- DRM Support, page 1-7
- ODE Playout for HDS, HLS, and HSS Formats, page 1-13
- Trick Mode Support for HDS, HLS, and HSS Playback, page 1-13

DRM Support

The VDS-OS supports the following digital rights management (DRM) types:

- Cisco Key Store 1.1 for Standard HLS, Encryption Type aes_128_cbc (HTTP only; HTTPS is not supported)
- Cisco VMS-KMS for Standard HLS, Encryption Type aes_128_cbc (HTTPS is not supported)
- Cisco VMS-KMS PlayReady for HLS, Encryption Type aes_128_ctr (HTTPS is not supported)
- Cisco VMS-KMS PlayReady for HSS, Encryption Type aes_128_ctr (HTTPS is not supported)
- Verimatrix VCAS 3.3 for Standard HLS, Encryption Type aes_128_cbc
- Verimatrix VCAS 3.3 PlayReady for HSS, Encryption Type aes_128_ctr

Encryption Enhancements

- Cisco Key Store Server Integration, page 1-7
- Cisco VMS-KMS Integration, page 1-8
- Verimatrix Key Server Integration, page 1-10

Cisco Key Store Server Integration

The VDS-OS integrates with the Cisco Key Store server to acquire content encryption keys for HLS ODE and playout.

The VDS-OS supports only offline VOD and VOD generated by the Cisco Transcode Manager (CTM). The Common Format (CF) VOD asset is on the NAS. ODE is used to play out content in HLS format.

Key Store Server Information

The Key Store is a safe-box component that is responsible for the allocation of content keys, secure storage, scalable provisioning, and key access control.

To run Key Store, enter the following command:

```
nds_service nds_keystore start
```

To stop Key Store, enter the following command:

```
nds_service nds_keystore stop
```
Cisco VMS-KMS Integration

The VMS-KMS Media Suite Key Management Server (VMS-KMS) supports key retrieval for both HLS and PlayReady.

The VDS-OS supports only offline VOD and VOD generated by the Cisco Transcode Manager (CTM). The DASH VOD asset is on the NAS. ODE is used to play out content in HLS/HSS format.

Note

The VMS-KMS key server maintains a database of all incoming URLs. Even if a request is refused with HTTP error response code 404, the key server saves the URL in the database. So if the key server receives the same request again, even if the new request has the correct configuration, it still responds with error code 404.

VMS-KMS Key Server Info

- Keyset Template: Each DRM type has its own keyset template, identified by a unique ID.
- Keyset: Each asset is linked to a keyset, identified by a unique ID. Multiple requests for the same asset by different clients receive the same keyset.
- Key: Each keyset contains a 16-byte key.
- Domain of the VMS server
- Username
- Password

Creating the Keyset Templates

You must create a keyset template for each encryption methodology, such as HLS or PlayReady. To do so, use the following procedure:

Step 1  On the VMS, select Key Management > Keyset Templates. The Manage Keyset Templates page is displayed.

Step 2  Click Add New. The Create New Keyset Template popup is displayed.

Step 3  Enter a Name of your choosing.

Step 4  Select a Key Provider.
Step 5  Select a DRM Type.

Step 6  Click Create to create the keyset template. The VMS automatically assigns a UUID (keyset template ID).

After you have configured the keyset templates on the VMS, you must add the content protection key profiles on the VOSM. Each origin service has its own Content Protection Settings. You can have multiple content protection key profiles, one for each key server and encryption type.

**VOSM Configuration for a VMS-KMS Key Profile**

The RSA private key file is needed to decrypt the key that is sent to the VDS-OS in the key server response. It is a part of the key profile for VMS-KMS. Therefore, you cannot add the RSA private key file until you have added the content protection key profile for VMS-KMS.

The RSA private key file is generated externally and received out-of-band.

Figure 1-1 shows the Creating New Content Protection Key Profile page for a VMS-KMS key profile.

**Figure 1-1   Creating New Content Protection Key Profile Panel for a VMS-KMS Key Profile**

For VMS-KMS, the content protection key profile requires the following information:

- Profile Name of your choosing
- Profile Type: **VMS-KMS**
- Protocol: **HTTP**

**Note**  HTTPS is not supported for VMS-KMS.

- Domain Name of the VMS server
- Port of the VMS server
- Auth Key
There is a VDS-OS client identity, created on the VMS, that calls the REST API with the username (UID) and password (PWD). The Auth Key is generated using base64 encoding of the username:password, and the word Basic is appended to the beginning of the generated Auth Key. For example:

Authorization: Basic Vk9TX0NMSUVOVDpwYXNzd29yZA==

- Key Set Template ID
- Key Provider Type:
  - HLS-AES-128
    VMS-KMS with HLS-AES-128 supports only HLS requests, which can be played using the QuickTime player on a MAC.
  - PR-AES-128
    VMS-KMS with PR-AES-128 supports only HLS and HSS requests. There is currently no player support for HLS with PlayReady encryption. HSS requests can be played using the PlayReady test application.
- RSA Private Key File
- RSA Private Key File Destination Name

Creating Video Assets In the VMS-KMS Key Server
You can create, edit, and delete video assets as bundle content in the VMS-KMS key server. After you have created a bundle, you can mark it as active, and it then is available for productization.

For detailed information about managing video asset bundles, see the information at the following URL:


where hostname is the domain name of the VMS server.

Verimatrix Key Server Integration
The VDS-OS integrates with the Verimatrix Video Content Authority System (VCAS) Key Server Version 3.3. The VCAS server provides key retrieval for HLS and Playready.

**Verimatrix HLS Playout**
The key request to the Verimatrix server is created based on the default request key template.

http://domain.com:1234/CAB/keyfile?r=1231234&t=VOD&p=0

Configure the domain and port using VOSM.

The response contains the key and the location header contains the key URI, which is sent to the end client for key retrieval. Additional information is appended to the location header, similar to the information that was sent in the request to the key server:

https://player.interface.dom/keyfile?r=1231234&t=VOD&p=0

Playout is supported only on the Verimatrix ViewRight player, which is available for download in the iOS App store.

**Verimatrix Smooth Playout**
The VDS-OS creates the key request to the Verimatrix server based on the default request key template:
Configure the domain port using VOSM. The response contains Playready DRM information. The body of the response might be empty. All of the parameters are received as custom headers in the HTTP response. Playout is supported on any browser that supports the Microsoft Silverlight plugin.

**VOSM Configuration for a Verimatrix Profile**

The RSA private key file is needed to decrypt the key that is sent to the VDS-OS in the key server response. It is a part of the key profile for Verimatrix. Therefore, you cannot add the RSA private key file until you have added the content protection key profile for Verimatrix.

The RSA private key file is generated externally and received out-of-band.

Figure 1-2 shows the Creating New Content Protection Key Profile page for a Verimatrix key profile.

**Figure 1-2 Creating New Content Protection Key Profile Panel for a Verimatrix Key Profile**

For Verimatrix, the content protection key profile requires the following information:

- Profile Name of your choosing
- Profile Type: Verimatrix
- Protocol: HTTP or HTTPS
- Domain Name of the Verimatrix server
- Port of the Verimatrix server
- Key Provider Type:
  - HLS-AES-128
    Verimatrix with HLS-AES-128 supports only HLS requests, which can be played using the QuickTime player on a MAC.
  - PR-AES-128
    Verimatrix with PR-AES-128 supports only HLS and HSS requests. There is currently no player support for HLS with PlayReady encryption. HSS requests with the Verimatrix PlayReady DRM can be played on any browser that supports the Microsoft Silverlight plugin.
- RSA Private Key File
- RSA Private Key File Destination Name

Adobe Access Encryption for Verimatrix
The VDS-OS supports Adobe Access encryption for Verimatrix. Adobe Access encryption requires access to an external Adobe Access license server, which you must configure in VOSM.

Step 1
Upload the following binary files for the license server and store them on the SE:
- Transport Certificate File
- License Certificate File
- Policy File
- Packager Certificate File

Step 2
Use the OS utility base64 command to encode each file in base64, then save the base64-encoded files on the SE.

Step 3
In the VOSM, select an origin service, then select Content Protection Setting in the left pane. The Content Protection Key Profiles panel is displayed.

Step 4
Click the Create New Content Protection Key Profile icon. The Creating New Content Protection Key Profile panel is displayed.

Step 5
Enter a Profile Name of your choosing, such as adobe-server.

Step 6
Select Profile Type Adobe License Server. The Adobe Access Files section is displayed.

Figure 1-3 Creating New Content Protection Key Profile Panel for Adobe License Server

Step 7
Enter the Adobe License Server URL - the URL of the license server.

Step 8
Choose the Transport Certificate File that you encoded for base64.

Step 9
Enter a Transport Certificate File Destination Name of your choosing.

Step 10
Choose the License Certificate File that you encoded for base64.

Step 11
Enter a License Certificate File Destination Name of your choosing.

Step 12
Choose the Policy File that you encoded for base64.
Step 13 Enter a Policy File Destination Name of your choosing.

Step 14 Choose the Packager Certificate File that you encoded for base64.

Step 15 Enter a Packager Certificate File Destination Name of your choosing.

Step 16 Enter the Packager Passcode that is associated with the Packager Certificate File. The passcode enables the VDS-OS to read the contents of the certificate.

Step 17 Click Submit to add the Adobe license server profile to the VOSM.

Step 18 Create a Verimatrix key profile (see the “VOSM Configuration for a Verimatrix Profile” section on page 1-11) with the following attributes:
   - Profile Name of your choosing, such as verimatrix-adobe3
   - Profile Type: Verimatrix
   - Protocol: HTTP
   - Key Provider Type: ADOBE-ACCESS
   - Adobe Access License Server: Profile Name of the Adobe license server (adobe-server in this example)

Step 19 Click Submit.

Step 20 Add the new key profile (verimatrix-adobe3 in this example) to the Asset Resolver file publish information to enable Adobe Access encryption for Verimatrix. For more information, see the “Asset Resolution” section on page 1-19.

ODE Playout for HDS, HLS, and HSS Formats

The VDS-OS supports the playout of ODE CF VOD content on the NAS for the following formats:

- **HDS**—HTTP Dynamic Streaming
  The VDS-OS support for HDS supports only 2-second fragment delivery.

- **HLS**—Apple HTTP Live Streaming
  The VDS-OS support for HLS supports both 10-second and 2-second fragment delivery, depending on the Asset Resolver rule configuration. The default setting is 10-second fragment delivery.

- **HSS**—Microsoft HTTP Smooth Streaming
  The VDS-OS uses the Asset Resolver file that is assigned to the origin service to map the requested ABR URL to the stored CF asset.

Trick Mode Support for HDS, HLS, and HSS Playback

The VDS-OS supports trick mode for HDS, HLS, and HSS playback. Trick mode, also known as trick play, provides functions such as fast-forward and rewind for digital video systems such as VOD.

To support trick mode, the VDS-OS stores Increased Digital Resolution (IDR) information, such as the byte offset and the size, in the SSIX.

The VDS-OS supports the following trick mode controls:

- **HDS**
  - Fast-Forward 2x
  - Fast-Forward 8x
- Rewind 2x
- Rewind 8x

- HLS
  - Fast-Forward 2x
  - Fast-Forward 4x
  - Fast-Forward 8x
  - Rewind 2x
  - Rewind 4x
  - Rewind 8x

- HSS
  - Fast-Forward 2x

**DASH-HELPER Client MPD for HDS**

A separate AdaptationSet is added to the DASH-HELPER client MPD to describe IDR-only ATS fragments. The AdaptationSet id attribute is set to i-frame.

```xml
<?xml version="1.0" encoding="utf-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns="urn:mpeg:DASH:schema:MPD:2011"
http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd" type="dynamic" mediaPresentationDuration="PT1098S" minBufferTime="PT4S"
profiles="urn:com:cisco:profile:common-format:2012">
  <Period id="1">
    <AdaptationSet mimeType="video/mp2t" codecs="avc1.676400" segmentAlignment="false"
startWithSAP="1" id="i-frame">
      <BaseURL>1/</BaseURL>
      <ContentComponent contentType="video" id="49"/>
      <SegmentTemplate timescale="90000" startNumber="27910"
media="$RepresentationID$/$Time$.ts">
        <SegmentTimeline>
          <S t="5028826778" d="180180" r="663"/>
        </SegmentTimeline>
      </SegmentTemplate>
    </AdaptationSet>
    <AdaptationSet mimeType="video/mp2t" codecs="mp4a.4c80" segmentAlignment="false"
startWithSAP="1" id="video_seg_2">
      <BaseURL>1/</BaseURL>
    </AdaptationSet>
  </Period>
</MPD>
```
Client Manifest for HDS

The i-frame streams are advertised in media with the type attribute set to video-keyframe-only.
Sample cMPD I-Frame Request for HDS

--2013-08-08 22:07:45--
Resolving localhost... ::1, 127.0.0.1
Connecting to localhost|::1|:80... failed: Connection refused.
Connecting to localhost|127.0.0.1|:80... connected.
Proxy request sent, awaiting response... 200 OK
Length: 5011 (4.9K) [application/octet-stream]
Saving to: `MPD'

Sample Manifest I-Frame Request for HDS

--2013-08-08 22:07:55--
http://vos.ipndvr1.com/1886/ndvr/ip/172.22.98.58/hds/Manifest.f4m
Resolving localhost... ::1, 127.0.0.1
Connecting to localhost|::1|:80... failed: Connection refused.
Connecting to localhost|127.0.0.1|:80... connected.
Proxy request sent, awaiting response... 200 OK
Length: 13969 (14K) [application/f4m+xml]
Saving to: `Manifest.f4m'

Sample 2x Fast-Forward I-Frame Requests for HDS

The following sample shows that a DCM-produced asset was used with one IDR (random access point) per 2-second ATS fragment.
Sample 8x Fast-Forward I-Frame Requests for HDS

The following sample shows that a DCM-produced asset was used with one IDR (random access point) per 2-second ATS fragment.

Sample 2x Rewind I-Frame Requests for HDS

The following sample shows that a DCM-produced asset was used with one IDR (random access point) per 2-second ATS fragment.

Sample 8x Rewind I-Frame Requests for HDS

The following sample shows that a DCM-produced asset was used with one IDR (random access point) per 2-second ATS fragment.
Admission Control

Different VDS-OS usage scenarios consume different amounts of CPU and RAM. Resources can also vary drastically between similar scenarios.

System resources are influenced by unforeseen runtime events, such as a memory leak. As the leaked memory increases, the number of sessions that VDS-OS can admit decreases.

The admission control algorithm dynamically adapts to different usage scenarios, coding changes, and runtime events, as well as conditions in the Web Engine itself.

VDS-OS supports admission control for both DASH and ODE assets. Other assets are not directly accounted for in admission control statistics and are not blocked by admission control.

To account for all of these factors, admission control admits sessions based on new system-wide values, such as CPU utilization and RAM utilization. That is, admission control functions similar to the Service Monitor/Service Router.

The Web Engine already provides an admission control mechanism that performs a number of process-specific control checks, including the Web Engine process memory threshold, file descriptor usage, sessions-per-second ramp up, and total sessions. VDS-OS accepts these Web Engine criteria as both valid and complementary to its own new system-wide criteria.

Table 1-1 shows the admission control criteria.

<table>
<thead>
<tr>
<th>Table 1-1</th>
<th>Admission Control Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Usage</td>
<td>System-Wide</td>
</tr>
<tr>
<td>Memory Usage</td>
<td>System-Wide</td>
</tr>
<tr>
<td>HTTP Sessions</td>
<td>Dynamic, based on system resources</td>
</tr>
<tr>
<td>FD Usage</td>
<td>None</td>
</tr>
<tr>
<td>Ramp Up</td>
<td>Dynamic, based on system resources</td>
</tr>
<tr>
<td>Media Assets</td>
<td>System-Wide</td>
</tr>
</tbody>
</table>

There is some overlap between admission control and load threshold, since they factor in memory and CPU utilization when determining whether the system is overloaded. However, admission control uses a weighted moving average that provides a more dynamic and proactive approach to determining overload conditions.
System-Wide Admission Control Criteria

Initially, system-wide criteria are based off of two resource dimensions (average CPU usage and average RAM usage) and across two major Web Engine resources (HTTP Sessions and Media Assets). VDS-OS uses these criteria to determine admission in two ways.

- Admission control imposes a global usage threshold. If either CPU usage or RAM usage for a specific Web Engine resource (such as Sessions or Assets) crosses a utilization boundary (for example, 90% CPU utilization), further resources are not admitted.

- Admission control derives RAM and CPU weights for each resource, based on current usage and on the current number of “active” resources. Admission control uses these values to predict how many new resources can be admitted.

For example, if Web Engine is running with 20% CPU utilization and 100 sessions, admission control might assume that each session requires 0.2% CPU. If the system is configured with a 90% global CPU threshold, admission control allows up to \[(90-20) / 0.2\] = 350 additional sessions.

For a low number of resources, resource weights might be too heavy. For example, a system with 20% memory usage and only one Media Asset might result in a 20% weight per asset. If the system had a 90% global threshold, that would mean that only three more assets could be admitted. To avoid that situation, admission control applies a “floor” threshold (for example, 50 assets), such that resource-weighting criteria are enabled only after the floor is exceeded.

Admission Control Averages

Admission control load averages are exponential moving averages (EMAs). EMAs are useful when calculating loads, and are used when calculating the familiar Unix load averages (for example, uptime). EMAs dampen sporadic instantaneous loads.

API Management

The VDS-OS provides HyperText Transport Protocol Secure (HTTPS) web services application programming interfaces (APIs) that comply with the REpresentational State Transfer (REST) standard, Java API for XML RESTful Web Services (JAX-RS), version JSR-311.

For more information, see the Cisco Videoscape Distribution Suite Origin Server Release 2.1.1 API Guide.

Asset Resolution

Each origin service is associated with an Asset Resolver configuration file. The Asset Resolver file contains a set of asset-based rules and actions for request processing. (All of the rules are asset-based, not user-based.) The Web Engine matches each incoming resource URL against the regex pattern defined in each rule in order to deliver assets to requestors.

The VDS-OS supports Asset Resolver for both DASH and ODE applications.

When creating an Asset Resolver, keep the following considerations in mind:

- The Asset Resolver performs CPU-intensive regular expression (regex) operations. Therefore, the number of rules and their order in the Asset Resolver file is critical. For better performance, we recommend that you use as few rules as possible, and make sure that the DASH or ODE rule is the first rule in the file.
The Asset Resolver uses the first matching regex rule. Therefore, rules that include specific regex patterns must come before more generic rules in the file.

For example, rules that include specific keywords must come before those with more generic ones.

For HLS, rules for stream manifests must come before rules for variant manifests.

If there is an overlap in URL namespaces for different types of assets (HLS Live, ODE VOD, recording, and so on), which require different Asset Resolver rules, you must create different origin servers to handle the different types of assets. You can then define the rules in separate Asset Resolver files and upload them to different origin services.

If the default Asset Resolver ODE rules match those of native VOD assets, you must either replace the default Asset Resolver or deselect it from the origin service.

For sample Asset Resolver files, including the default Asset Resolver file, see the “Creating Asset Resolver Files” section on page 1-1.

**Asset Resolution for ODE**

When ODE is enabled, the SetTransformResource Directive contains SourceInfo and PublishInfo.

- SourceInfo provides parameters for the source of the transform.
- PublishInfo provides the publish manifest URL for client.

When an incoming HDS or HLS media request URL matches the regex, the Asset Resolver creates an ODE asset, based on the parameters specified in SourceInfo and PublishInfo. The mpdURL is sent to the DASH media application, and a DASH asset is also created.

- For HLS media assets, the VDS-OS supports either 2-second or 10-second segment durations. The Asset Resolver uses the segmentDuration to specify the expected segment duration, which must match the segment duration of the video source for the DASH application.
- For HDS media assets, the VDS-OS supports only 2-second segment durations.

**Default Asset Resolver File**

The VDS-OS provides a default Asset Resolver. The VOSM automatically assigned the default Asset Resolver to an origin server when it is created.

If your URL format and asset storage structure differ from those in the default Asset Resolver, you must customize the Asset Resolver accordingly and reapply it to the origin service.

You can also assign a different Asset Resolver file as the default. For more information, see the “Set a File as the Default” section in the *Cisco Videoscape Distribution Suite Origin Server Release 2.1.1 API Guide*.

---

**Note**

The current default Asset Resolver file assumes that the CF content is stored on the NAS in the following directory structure for HLS playout: `/sharedDirectory/AssetDirectory/`

This is to identify the asset structure in variant manifest and stream manifest URLs.

Other NAS structures require a separate Asset Resolver file.

HDS and HSS play out correctly for all NAS structures with the default Asset Resolver file.
Cache-Control Enhancements

- Cache-Control Headers Support, page 1-21
- Support for If-Modified-Since for Native ABR for HLS Live and for HDS, HLS, and HSS VOD, page 1-23
- Support for If-Modified-Since for ODE Playout from CF VOD on the NAS, page 1-25

Cache-Control Headers Support

The cache-control general-header field is used to specify cache directives that must be obeyed by all caching mechanisms along the request/response chain. The directives specify behavior that prevents caches from adversely interfering with the request or response. These directives override the default caching algorithms. Cache directives are unidirectional, in that the presence of a directive in a request does not imply that the same directive is to be given in the response.

The VDS-OS supports the following tools to set the cache-control headers in the HTTP response for contents.

Note
Only the max-age response cache directive is currently supported.

<table>
<thead>
<tr>
<th>Cache-Control Directive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Cache-Control for ABR VOD Asset</td>
<td>The default cache-control max-age is based on the Content Expiry field on the Origin Service General Settings page.</td>
</tr>
<tr>
<td>Set Cache-Control for Asset Class Using Asset Resolver</td>
<td>You can override the default cache-control directives using an Asset Resolver file with rules for asset classes.</td>
</tr>
<tr>
<td>Default Cache-Control for HLS Live Asset</td>
<td>For HLS Live assets, you must take special care when setting the cache-control max-age for manifest files, which are updated frequently. The following values are used for the different HLS Live files:</td>
</tr>
<tr>
<td></td>
<td>• Stream Manifest File—The stream manifest file is updated every time a segment is added, which is based on the Capture Resource Arrival Rate setting. The VDS-OS sets the Expiry header, based on when it expects the next update to take place, by adding the Arrival Rate to the Last Modified Time.</td>
</tr>
<tr>
<td></td>
<td>• Variant Manifest File—The variant manifest file is updated only when a stream is added or removed due to a failure. The VDS-OS sets the Expiry header by adding the Arrival Rate to the Current Time.</td>
</tr>
<tr>
<td></td>
<td>• Segment File—The life of a live segment file is the size of the Time Shift Window. The VDS-OS sets the Expiry header by adding the Time Shift Window to the Last Modified Time.</td>
</tr>
</tbody>
</table>
The following table shows the expected **max-age** in the cache-control header, based on the type of playout.

<table>
<thead>
<tr>
<th>Playout Type</th>
<th>Variant Manifest</th>
<th>Stream Manifest</th>
<th>Fragment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Republish Live (NAS)</td>
<td>Arrival rate</td>
<td>Arrival rate—time on disk</td>
<td>Remaining DVR window</td>
</tr>
<tr>
<td>Republish Live (InMemory)</td>
<td>Arrival rate</td>
<td>Arrival rate—time in disk</td>
<td>Remaining DVR window</td>
</tr>
<tr>
<td>Passthrough Live (NAS)</td>
<td>Arrival rate</td>
<td>Arrival rate—time on disk</td>
<td>Remaining DVR window</td>
</tr>
<tr>
<td>Passthrough Live (InMemory)</td>
<td>Arrival rate</td>
<td>Arrival rate—time in disk</td>
<td>Remaining DVR window</td>
</tr>
<tr>
<td>HLS VOD (Completed recording)</td>
<td>Content expiry</td>
<td>Content expiry</td>
<td>Content expiry</td>
</tr>
<tr>
<td>HLS VOD (InProgress)</td>
<td>Arrival rate</td>
<td>Arrival rate—time on disk</td>
<td>Content expiry</td>
</tr>
<tr>
<td>ODE VOD (Through content expiry)</td>
<td>Content expiry</td>
<td>Content expiry</td>
<td>Content expiry</td>
</tr>
<tr>
<td>ODE VOD (Through Asset Resolver)</td>
<td>Asset Resolver configured value</td>
<td>Asset Resolver configured value</td>
<td>Asset Resolver configured value</td>
</tr>
</tbody>
</table>

**Note** When both configurations are present, the Asset Resolver configuration has priority.

For more information about configuring the **max-age** for a cache-control header using the VOSM, see the “General Settings” section on page 1-3.

To override the default cache-control directives using an Asset Resolver file, specify the **max-age** in seconds, for a set of assets that match a set of regular-expression (regex) rules. The **max-age** can apply to an entire origin service (that is, it applies to all requests) or it can apply to a subset of requests, such as m3u8 index files or HLS fragment requests.

**Using Asset Resolver To Set Max-Age for an Origin Service Example 1**

```xml
<Rule_Actions>
  <Rule_AssetResolve nametag="global_caching_headers_rule" matchGroup="grp1, grp2" protocol="http">
    <SetMetaData>
      <DeliveryHeader>
        <SetCachingHeader>
          <Cache-Control>
            <Max-Age>300000</Max-Age>
          </Cache-Control>
        </SetCachingHeader>
      </DeliveryHeader>
    </SetMetaData>
  </Rule_AssetResolve>
</Rule_Actions>
```

**Using Asset Resolver To Set Max-Age for an Origin Service Example 2**

**Note** With the current Asset Resolver design, if you set **max-age** for HLS .ts alone, the same **max-age** also applies to all other HSS and HDS fragments. If that is unacceptable, you must set a specific **max-age** for HDS and HSS fragments.
Support for If-Modified-Since for Native ABR for HLS Live and for HDS, HLS, and HSS VOD

The If-Modified-Since HTTP header is sent by end-clients or caches to revalidate the request. The server responds with one of the following codes:

- **200 (OK)**—If the page has been modified
- **304 (Not modified)**—If the page has not been modified
The If-Modified-Since HTTP header is supported by the VDS-OS for the following types of ABR playout:

- Native HLS Live manifest and segments (pass-through or republish with NAS or In-Memory)
- Native HLS time-shift TV (TSTV) asset
- Native HLS, HSS, HDS VOD asset

An If-Modified-Since request can be sent in any type of HTTP requests, such as a GET request, a HEAD request, or a Range request. For each request, the VDS-OS compares the date and time specified in the If-Modified-Since to the Last-Modified date and time of the resource, and sends the appropriate 200 or 304 response.

**Sample If-Modified-Since HTTP Header 1**

The following header shows an HTTP GET request for which the If-Modified-Since time is greater than the Last-Modified time for the HLS segment file.


HTTP/1.1 304 Not Modified
Connection: Keep-Alive
Keep-Alive: timeout=15
Accept-Ranges: bytes
Cache-Control: max-age=500
Last-Modified: Wed, 22 May 2013 17:52:28 GMT
Content-Type: video/MP2T
Server: Cisco-VOS
Content-Length: 0
Date: Wed, 22 May 2013 17:46:58 GMT

**Sample If-Modified-Since HTTP Header 2**

The following header shows an HTTP GET Range request for which the If-Modified-Since time is greater than the Last-Modified time for the HLS stream manifest file.

dbgshell# curl -i --header 'Range:bytes=10-20' --header 'If-Modified-Since: Tue, 21 May 2014 11:16:00 GMT' http://vos.ruixli.com/ruixli_hls/hls/high3000.m3u8

HTTP/1.1 304 Not Modified
Connection: Keep-Alive
Keep-Alive: timeout=15
Accept-Ranges: bytes
Cache-Control: max-age=4
Last-Modified: Wed, 22 May 2013 17:48:52 GMT
Content-Type: application/x-mpegURL
Server: Cisco-VOS
Content-Length: 0
Date: Wed, 22 May 2013 17:48:58 GMT

**Sample If-Modified-Since HTTP Header 3**

The following header shows an HTTP GET request for which the If-Modified-Since time is less than the Last-Modified time for the HLS manifest file.

dbgshell# curl -i --header 'If-Modified-Since: Tue, 21 May 2013 11:16:00 GMT' http://vos.ruixli.com/ruixli_hls/hls/high3000.m3u8

HTTP/1.1 200 OK
Connection: Keep-Alive
Sample If-Modified-Since HTTP Header 4

The following header shows an HTTP GET request for which the If-Modified-Since time is greater than the Last-Modified time for a large HSS video file. (HSS audio and video fragment requests with If-Modified-Since are handled the same way.)

dbgshell# curl -I --header 'If-Modified-Since: Wed, 30 May 2012 21:45:00 GMT' http://vos.ruixli.com/hssvod/BigBuckBunny_477.ismv

HTTP/1.1 304 Not Modified
Connection: Keep-Alive
Keep-Alive: timeout=15
Accept-Ranges: bytes
Content-Length: 47374020
Last-Modified: Sat, 29 Sep 2012 00:37:04 GMT
Content-Type: application/octet-stream
Server: Cisco-VOS
Date: Thu, 30 May 2013 22:44:24 GMT

Support for If-Modified-Since for ODE Playout from CF VOD on the NAS

For ODE HDS, HLS, and HSS resources from a CF VOD asset on the NAS, the Last-Modified time for the CF asset is compared to the If-Modified-Since time.

External Storage Devices—NAS

Network Attached Storage (NAS), a file-level storage connected to a network, is supported by VDS-OS. NAS file-systems can be accessed using the NFS protocol.

A NAS may be traditional or clustered. A clustered NAS is a NAS that uses a distributed file-system running simultaneously on multiple servers. The key difference between a clustered and traditional NAS is the ability to distribute (also known as stripe) data and metadata across the cluster nodes or storage devices. Clustered NAS, like a traditional one, still provides unified access to the files from any of the cluster nodes, unrelated to the actual location of the data. Any of the storage servers in the cluster can be used to mount a file-system volume.

Note

Performance and efficiency of the VDS-OS system may suffer if network storage is accessed using a high-latency or low-bandwidth wide-area network (WAN). To monitor storage latency and bandwidth, use the diagnostic counters and statistics pertaining to network storage.

The following assumptions are made about network storage topology for VDS-OS:

- There may be multiple independent NAS storage clusters in a VDS-OS deployment, spread across different locations.
- Storage clusters and VDS-OS nodes accessing storage within those clusters may be in different locations.
• Network storage cluster in one location may be accessible from VDS-OS nodes that are both co-located and in other locations.

| Note | Network traffic performance can be impacted by too small a value for the TCP parameter: net.inet.tcp.rexmit_slop. If it is determined that network throughput performance is impacted, the net.inet.tcp.rexmit_slop value on the Network Storage Share server should be reviewed. |

### High Availability—Resiliency and Redundancy

A VDS-OS that is designed with full redundancy and no single point of failure includes redundant VOSMs and Service Routers.

### Service Engine Redundancy

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 origin service. All the existing sessions on the failed Service Engine terminate and the affected end users must re-request the content.

### Service Router Redundancy

The VOSM 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-OS network; however, you can have any number of VOSMs operating in standby to provide redundancy and failover capability.

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

The VOSM design principle is that the management device is never in the service playout path. When the VOSM fails, the rest of the VDS-OS continues to operate. A VOSM failure does not affect any services delivered to end users, and all content sourcing continues. The only negative effect is that the administrator cannot change configurations or monitor the VDS-OS. As soon as a failure to connect to the VOSM is noticed, the administrator can activate the standby VOSM. For information on making the standby VOSM the primary VOSM, see the “Changing a Standby to a Primary VOSM” section on page 1-12.

### Operational Enhancements

- Cross-Domain Policy File for HDS Playback, page 1-26
- Customized Asset URLs, page 1-27

### Cross-Domain Policy File for HDS Playback

The cross-domain policy file is an XML document that grants permission to a web client, such as the Adobe Flash Player or Adobe Reader, enabling the client to handle data across multiple domains.
When a client hosts content from a particular source domain, and that content directs requests toward a domain other than its own, the remote domain must host a cross-domain policy file in order to grant access to the source domain and allow the client to continue with the transaction. Cross-domain policy files grant read access to data, permit the client to include custom headers in cross-domain requests, and grant permission for socket-based connections.

During playback, the HDS player first initiates a request to the VDS-OS for the cross-domain policy file. To upload a cross-domain policy file for each origin service, select the Cross-Domain File Registration option on the VOSM System > Configuration page.

### Customized Asset URLs

The VDS-OS keeps a table of active assets for in-media applications (HDS, HLS, HSS, and DASH). Each asset has a unique asset URL that helps identify the asset to which the client request URL should attach.

The Asset Resolver `assetUrl` attribute identifies the asset URL, which is used as the base URL for the asset or profile and ODE. You can add an `assetUrl` attribute for `PublishInfo` to specify the asset URL for the client request URL.

Note

The generated asset URL must be unique for each asset.

For ODE, part of the asset URI is received as the `vm_id`. ODE then uses the `vm_id` to generate the variant manifest file.

### Sample URL for HLS

Given the following incoming manifest URL:

http://fqdn/rec_id.m3u8

The Manifest URL is:

http://fqdn/rec_id.m3u8

The AssetUrl URL can be:

http://fqdn/rec_id

The prefix ID `rec_id` is passed to the ODE, which generates the variant manifest with the following content:

rec_id/bitrate.m3u8

The client’s profile manifest URL is:

http://fqdn/rec_id/bitrate.m3u8

The profile manifest file contains the following media file:

`bitrate/number.ts`

The client’s media data URL is:

http://fqdn/rec_id/bitrate/number.ts

### Sample Asset Resolver Rule for HLS Media Data File for the Sample URL

```
<SetTransformResource matchRegex = "\.(.*)/\.(.*)/\.(.*)/\.(.*)\.ts">
```
Sample URL with Current VDS-OS Supported

Given the following incoming manifest URL:
http://fqdn/rec_id/anything/index.m3u8

The Manifest URL is:
http://fqdn/rec_id/anything/index.m3u8

The AssetUrl URL can be:
http://fqdn/rec_id/anything

No specified prefix ID is sent to the ODE. The ODE generates the variant manifest file, which contains the following content:
bitrate.m3u8

The client’s profile manifest URL is:
http://fqdn/rec_id/anything/bitrate.m3u8

The profile manifest file contains the following media file:
bitrate/number.ts

The client’s media data URL is:
http://fqdn/rec_id/anything/bitrate/number.ts

Sample URL for HDS

Given the following incoming manifest URL:
http://fqdn/rec_id.m3u8

The Manifest URL is:
http://fqdn/rec_id.f4m

The AssetUrl URL can be:
http://fqdn/rec_id

The prefix ID /rec_id is passed to the ODE, which generates the manifest with the following media_url template:
rec_id/media_file_template_prefix

The client’s media data URL is:
http://fqdn/rec_id/media_file_name
Display Usage for TMPFS and NSS Usage

The `show vos usage` command displays usage information for the VDS-OS, including:

- Network Storage Shares (NSS) usage for every Network File System (NFS) in the SE
- Temporary file storage (TMPFS) usage

For more information, see the description of the `show vos` command in the *Cisco Videoscape Distribution Suite Origin Server Release 2.1.1 Command Reference*.

Alarm Enhancements

Alarms 560015 (`nas_threshold`) and 1100002 (`channel_failure`) have been added.

Alarms 9000015 (`WritingToNasHasFailed`) and 1100001 (`channel_partial_failure`) have been enhanced.

For more information, see the alarm descriptions in the *Cisco Videoscape Distribution Suite Origin Server Release 2.1.1 Alarms and Error Messages Guide*.

Origin Services

Content origination is performed in the context of *origin services*. An origin service is a construct that is used to map the content and channels to a set of Service Engines that work together to originate content for the content distribution systems. Multiple origin services can coexist in the same VDS-OS system. The origin services are used to capture content (for example, fragments, streams, and so on) for playout to CDNs.

**Note**

An origin service is restricted to a single location.

The built-in service routing and service definition capabilities of VDS-OS enable operators to manage services instead of servers. VDS-OS is used to provision, manage, and operate origin servers as a single unified service that has built-in fault tolerance and proximity-based server selection functions.

Following are the different properties of the origin services:

- **Service identification**—Each origin service is uniquely identified with an origin service name and a Service Routing Domain Name (SRDN), which is a fully-qualified domain name (FQDN). The names and the SRDNs must not overlap with other services in the system.

  The URL for all of the resources delivered from this service have the SRDN as the domain part. The template for the URL is: `http://<SRDN>/<Content_Specific_Identifiers>`

  For example, if the SRDN is `origin.tveverywhere.net`, then following are some URLs of resources that might be delivered from this service:

  - `http://origin.tveverywhere.net/nbc/nbc1/live.m3u8`
  - `http://origin.tveverywhere.net/nbc/nbc1.isml/Manifest`

- **Content types**—Any static HTTP resource that conforms to the HTTP specification can be delivered from an origin service. However, the service is optimized around Adaptive Media Streaming (HDS [VOD only], HLS, and HSS [VOD only]). The Media Delivery Applications are used to deliver the ABR content. For more information about Media Delivery Applications, see the “Media Delivery Applications” section on page 1-33.
Feature Support

Chapter 1      Product Overview

Note Each ABR content (VOD, live, or Live-to-VOD) is treated as a resource.

- Content Resources—Content resources consist of the following:
  - Capture Resource—Defines input streams the system should expect for a channel of every ABR type. VDS-OS supports HLS capture resources. The format-specific capture sources for the same channel have the same channel identification (there is a base URI that points to the channel).
  - Linear Publish Resource—Publishes the captured channel to the CDN systems. Multiple publish resources can use a common capture resource. For example, HLS Live streams for a channel (capture resource) can be published to the end users with two publish resources (for example, a smart-phone profile and a tablet-profile). The publish resource can use all or a subset of streams defined in the capture resource. The live channel publish resources also have settings that define the in-session time-shift windows and start and end times.
  - VOD Publish Resource—Publishes VOD content that is prerecorded into the authoritative storage (Network Storage Shares). There is no special configuration for this, the system automatically creates the VOD publish resource based on the client request.
  - Time-Shift Recording and Publish Resources—Records and publishes content in support of TSTV services. The TSTV publish resources are created based on operator-specified schedules (using the Capture Schedule XML file) or can be created dynamically on the Acquisition nodes based on a control command from an external session manager. When the end-time of a capture event has been reached, the content produced during the capture is converted to a VOD asset that can be delivered by the VOD publish resources. These schedules may change, so there is a polling interval specified to refresh the Capture Schedule XML file periodically.

- Multi-Tenancy—Multiple origin services can be provisioned in the system. The Service Engines assigned to an origin service can be shared with other origin services. The Service Engines are inherently multi-tenant and can host multiple origin services concurrently. The number of origin services that are supported by a Service Engine are dependent on the peak origination workflow requirements across the different services.

- Service participants and routing—Service Engines participate in the content origination workflows. One or more Service Engines can be assigned to an origin service.
  The Service Routers are delegated to be the authoritative DNS server for the SRDN. Service Routers are primarily used to select origin server instances based on various policies such as proximity, server load, and availability thresholds.

Note VDS-OS supports only DNS-based routing.

- CDN integration—VDS-OS can work with any CDN that can download content using HTTP as specified in RFC 2616. The CDN nodes do not know the difference between an origin service and an external origin server; both of them would export the same protocol interface.

- Playout QoS settings—During content playout, all resources played out from the VDS-OS system can be marked with specific DSCP markings. A configuration for QoS setting can be specified for each origin service.

- Asset Resolver rules—The Asset Resolver rules can help resolve incoming fragment, manifest, and other types of requests to the publish resource that can deliver the request. Given a URL, the action to be performed and the parameters associated with that action can be specified in the Asset Resolver XML file.
Live Channels

Each origin service can support the playout of one or more live channels. Each live channel describes the following:

- ABR format-specific publish resources
- Capture resources
- Pointer to the live Capture Schedules (for live-to-VOD conversion)
- Acquisition nodes that can acquire the live streams from external encapsulators

Each live channel is configured with at least one Acquisition node.

Network Storage Shares

The Network Storage Shares (NSS) define the external storage directories that are mounted by the Service Engines by way of the origin services. Each share is mapped to a mount point name that can be derived from the incoming request URLs. The NSSs are the export file systems from the NAS servers.

**Note**

At least one Network Storage Share must be assigned to an origin service and a live channel.

Content Sourcing

Content sourcing describes the process of acquiring content in the origin service. The sourcing setup workflows can differ based on the solution (linear, VOD, and TSTV).

Table 1-2 describes the different potential sources of content and the protocols and transports used for the different scenarios.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Sources</th>
<th>Signaling Protocol</th>
<th>Data Transport</th>
</tr>
</thead>
<tbody>
<tr>
<td>Managed linear capture with external</td>
<td>External encapsulator</td>
<td>Not applicable</td>
<td>HLS</td>
</tr>
<tr>
<td>encapsulator (linear, TSTV)</td>
<td></td>
<td></td>
<td>HTTP POST (1 transaction per segment)</td>
</tr>
<tr>
<td>VOD content origination (NSS read)</td>
<td>External storage servers</td>
<td>NFS</td>
<td>NFS</td>
</tr>
</tbody>
</table>

VOD content origination (NSS read) is sourced from external storage servers and uses NFS as the signaling protocol and as the data protocol.

VOD content is packaged and stored in its native format (also known as post-encapsulated format) into Network Storage Shares along with the control and distribution metadata. The operator’s content management system is responsible for storing the content. The Service Engines can mount the NAS volumes and deliver the requested objects from the stored containers.

Live content is encapsulated for each playout technology and received by the VDS-OS Acquisition nodes by way of HTTP POST.

VDS-OS Acquisition nodes record content and publish it.
Chapter 1      Product Overview

Content Playout

The playout process of the VDS-OS system delivers content to requesting clients (CDN nodes). All Service Engines that are part of the origin service are able to playout VOD content as long as the content is completely available as a recording. Live content is streamed by way of linear capture resources. For content playout, the origin service must be configured and the Service Router must be notified about the origin service and its participating Service Engines by way of the Coverage Zone file.

VOSM

The VDS-OS consists of a Virtual Origin System Manager (VOSM), one or more Service Engines (SEs), and one Service Router (SR). For full redundancy, a VDS-OS would include an additional VOSM and Service Router.

The VOSM is a secure Web browser-based user interface. It is a centralized system management device that allows an administrator to configure, manage, and monitor origin services as well as all devices in the VDS-OS.

APIs are provided for backoffice integration with the VOSM.

Authentication, Authorization, and Accounting

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

User accounts and groups can be added to the VOSM 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 VOSM provides its own authentication server.

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

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

Device Management

The VOSM 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 VOSM, 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 VOSM also provides an automated workflow to apply software upgrades to the devices in a device group.
Origin Services Management

The VOSM provides the configuration and monitoring of origin services, which defines how content is sourced, recorded, and played out. The VOSM provides the Service Engines with information about the origin services and which Service Engines are participating in the origin service.

Service Engine (SE)

The Service Engine forms the basic building block for most of the content origination and distribution services. The Service Engine handles content sourcing, content distribution within the VDS-OS, and content playout.

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.

Media Delivery Applications

Media Delivery Applications are software modules that are implemented on top of the Web Engine. The Media Delivery Applications perform the following functions:

- Source content and recording incoming linear streams
- Locate requested content from a stored location based on the requested URL
- Implement delivery technology specific behaviors (for example, fragment chaining, manifest creation, and so on)
- Implement Live-to-VOD content creation

The Media Delivery Application integrates and abstracts the functions such as content sourcing, recording, re-encapsulation, and playout. VDS-OS supports the following Media Delivery Applications:

- HDS Media Application (VOD)
- HLS Media Application (VOD, Live, and Live-to-VOD)
- HSS Media Application (VOD)

Linear Stream Acquisition

Linear stream acquisition is a function that is used to acquire the linear streams from the encapsulator.

Capture Controller

The Capture Controller is a software subsystem that controls the Web Engine Media Delivery Applications. It is used for all of the live content captures, including live-to-VOD conversion. The Web Engine implements the recording and playout of these captures.

The Capture Controller performs the following functions:

- Channel Management—Processes live channel configuration received from the VOSM and maintains the states of the live channels and capture resources.
- Interface Monitor—Examines the interfaces to obtain details about on-going and completed capture sessions.
Feature Support

- Capture Scheduler—Stores live and live-to-VOD events for all channels based on the Capture Schedule XML file and schedules these events to the Web Engine.
- Capture Life Cycle Management—Maintains the state of the live sessions, as well as live-to-VOD recording sessions.
- Channel Operation—Handles control messages to start, stop, or restart a publish resource or a capture resource.

Web Engine

The Web Engine is used to implement the recording and playout of the recorded assets.

All HTTP requests that are redirected to a Service Engine by the Service Router are handled by the Web Engine. The Web Engine, using HTTP, can serve the request from locally stored content in the VDS-OS.

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

The Web Engine supports the following:
- Optimization for small content objects
- Optimization of Adaptive Bitrate Streaming for Move, Apple iPhone, and Smooth HD
- 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, HTTP POST, and HEAD request methods.

Service Router (SR)

The Service Router for the VDS-OS system is used to load-balance incoming requests to different origin server instances. It is also used to provide redundancy within a VDS-OS service location as well as select between multiple sites.

The Service Router can be configured only as the Request Routing Engine.

In describing the Service Router functions, the term “client” refers to the CDN Acquisition node.

Request Routing Engine

*Request Routing Engine* is used to load-balance incoming requests to different origin server instances. It is also used to provide redundancy within a VDS-OS Service location as well as select between multiple sites.
The Request Routing Engine supports DNS-based routing and RFQDN redirection. All the Service Engines registered and activated in the system send heartbeat messages to the Service Router for load monitoring purposes.

**Routing Redirection**

The Request Routing Engine uses DNS-based redirection as the routing method.

When DNS-based routing is configured, the Service Router routes the request based on the DNS proxy that is used by the downstream CDN Acquisition nodes. The IP address of the actual CDN node is not carried in the DNS request.

**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.

DNS-based routing is enabled by default. When DNS-based redirection is enabled, the DNS proxy contacts the Request Routing Engine to resolve the requested URL, 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 CDN Acquisition node 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 is a certain side effect in adopting this approach. When creating the Coverage Zone file, the IP address of the DNS proxy needs to be used for the CDN Acquisition node IP address range.

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 voslive.com domain as the domain used to redirect all requests to:

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

**Routing Methods**

The Request Routing Engine chooses the best Service Engine based on whether the Service Engine is participating in the origin service for which the origin server matches that of the requested domain, and whether the Service Engine is assigned to serve the CDN Acquisition node’s network region, as defined in the Coverage Zone file.
If the CDN Acquisition node’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**
- **Zero-IP Based Configuration**
- **Last-Resort Routing**
- **Service Aware Routing**

**Note**
The keepalive messages between the Service Router and Service Engine are transmitted and received on port 2323. However, the software interoperates 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.

**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, 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.

**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 content origin (if Enable Origin Server Redirect is enabled) for one of the following conditions:

- All SEs in the origin service have exceeded their thresholds
- All SEs in the origin service are unavailable or no SEs are assigned to the origin service
- Client is unknown

**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 a content origin 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 in order 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 Origin Service 2.1.1 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.

**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 not exceeded its threshold as configured. See the “Setting Service Monitor Thresholds” section on page 1-52 for more information.

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.

**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.
Request Routing Engine Workflow of Coverage Zone Routing

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.

VDS-OS 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.

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-OS serves a number of different domains (configured as origin services), but uses the same set of SEs for the different origin services (domains). By enabling the unified routing table in this scenario, the memory usage on the SR is reduced.

If the VDS-OS 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.

Coverage Zone File

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.

## Device Modes

The VDS-OS uses the device modes (VOSM, Service Engine, and Service Router) and applies the software functions that are required to perform content origination. The device mode configuration determines the set of software functions that can be activated in the device.

- **Virtual Origin System Manager**—Manages all the devices (Service Engines and Service Routers). In addition to the element management functions, the VOSM is also used to configure the system topology and services required for content origination. The VOSM supports a graphical user interface (GUI) and Web Services APIs for provisioning and monitoring VDS-OS. The VOSM provides device and service alarm information and reports statistical information on VDS-OS functionality.

- **Service Engine**—Provides all HTTP content origination functions, satisfying content requests from the CDN or directly from clients, sourcing live content from HTTP ABR encoders and encapsulators, and sourcing VOD or recorded content from the operator’s NAS subsystem. The Service Engine provides necessary ABR segmentation and fragmentation to satisfy requests for HTTP ABR content.

- **Service Router**—Provides optimal origin server selection and load-balancing enabling a highly resilient origin solution. Resolves DNS queries for the origin service domain names. The queries are resolved to the IP address of the Service Engines (SEs). The Service Router (SR) takes into account the device load, availability, and proximity of the SEs (with reference to the requesting DNS proxy) when resolving the DNS queries.

VDS-OS is implemented as a set of networked devices organized in a well-defined topology. The topology consists of a single location.

## System Topology

The VDS-OS is a single-tier solution. The topology is defined as a set of stand-alone locations that are not inter-connected by way of the parent and child relationship.

Locations imply the topological proximity of the nodes that belong to that location. The relationship among locations guides the flow of content between the locations. Each location is configured with a unique-name that describes the location.

During the registration and activation process, all devices are assigned to a location. A device cannot be part of multiple locations.

For origin services, the locations are organized as a set of stand-alone locations. Each location can correspond to a data center (also known as a super headend, or SHE) location and these locations are not interconnected in the topology. There is no inter-location content movement through the VDS-OS nodes. Origin services cannot span more than one location in this release.

Based on the topology, different network zones can be created (manually) in the Service Router's Coverage Zone file to indicate the subset of nodes that are exposed to the downstream CDN nodes and to associate a location based on the subnet and proximity of the downstream CDN nodes.
Figure 1-5 shows the major elements of a VDS-OS system. How content flows, from source to recording within the VDS-OS, to playout to CDNs, is dictated by the defined origin services. Live playout flows directly from the Acquisition nodes to the Playout nodes. VOD content is retrieved from the Network Storage Shares by the Playout nodes.

**Figure 1-5 High-Level View of the Cisco VDS-OS**

Service Engines have two roles:
- Acquisition nodes
- Playout nodes

**Acquisition Nodes**
Acquisition nodes receive content from live sources and facilitate redundancy within the origin tier for live and recording content.

Linear content is acquired by the Acquisition nodes from external encapsulators. The acquired linear content can be received into the Network Storage Share devices associated with the origin service. At least one Acquisition node must be assigned to a live channel.

VOD content is not acquired by the Acquisition nodes directly. Instead, the Cisco Media Suite (CMS) directly ingests the content into the Network Storage Shares.

**Note**
At least one Network Storage Share must be assigned to an origin service and a live channel.

**Playout Nodes**
All nodes in an origin service that deliver content to the CDN are Playout nodes. An Acquisition node can also be a Playout node; however, the live channel redundancy capability is reduced. The Playout nodes are defined in the Coverage Zone file for the Service Router that resolves the DNS requests for that domain. The Playout nodes also mount the Network Storage Shares and can play the content out that is completely received and available in the storage system. For all content that is not completely received
(for example, Live-to-VOD captures that are not complete), the Playout nodes dynamically fetch the content from the Acquisition node. The set of Acquisition nodes for a Live-to-VOD capture and live channels are well known to all Service Engines participating in the origin service.

Virtual Machine (VM) Support

An SE can boot up as a VM in any hardware platform that meets the minimum resource requirements. VDS-OS support for VMs requires the following software programs and releases:

- VMware ESX hypervisor version 5.1 or later, running on the server
- VMware vCenter version 5.0 or later
    - We recommend that you use vCenter to deploy VMs. Using vCenter to configure a static IP address, a VOSM primary management IP address, and other basic VM parameters prior to deployment ensures that the VM will be functional when it first boots up.
- VMware vSphere version 5.0 or later
    - You can use vSphere to monitor and control basic features on the VM, but advanced features require vCenter.

Use the Open Virtual Appliance (OVA) to deploy the VM.

For more information about deploying a VM, including configuring the UCS B200 for load balancing, see the “Deploying the VDS-OS As a Virtual Machine” chapter.

Provisioning of New Features

- You can configure VDS-OS with multiple SEs.
- The Service Router coverage zone file is configured to include the SEs that are used to receive the request from the end-clients.
- The Asset Resolver configuration file includes a special directive that can derive the URL for the recordings or the URL for the recorder discovery function.
Figure 1-6 shows the sequence of operations to provision VDS-OS features.

**Figure 1-6  Provisioning Sequence of Operations**