Adobe® Primetime

Origin Server Getting Started Guide

Updated 3 February 2014

Contents

Origin Server Getting Started Guide

Overview

Adobe Primetime Origin Server serves HTTP streaming files to HTTP Dynamic Streaming (HDS) clients or HTTP Live Streaming (HLS) clients. It accepts HDS or HLS content from Primetime packagers or encoders and serves it to HDS or HLS clients over HTTP.

Origin Server can accept HDS and HLS input from Primetime packagers. Primetime Origin Server accepts HTTP PUT requests from one or more Primetime packagers to publish content. The Origin Server is used to serve out HDS and HLS artifacts.

The distribution file contains default configuration settings that allow Origin Server to start up, accept HDS fragments from the packager, and publish them.

What's new in Origin Server, Version 1.2

- *MP4 JIT Packaging*
- *Access Plugin for remote mp4*
- *Web VTT support*
- *JIT Audio Only*
- *Parsing DPI cues*

Setting up Origin Server

The distribution file contains default configuration settings that allow Origin Server to start up, accept HDS fragments from the packager, and publish them.

Note: *Origin Server requires Oracle Java Runtime Environment (JRE) 1.6 or later. Java SDK is required for the JConsole access.*

Default stream hierarchy to support multi-tenancy

Origin Server has a two-level hierarchy for grouping streams. At the top level, there is a container directory that contains one or more container sub directories. The name of the directory identifies the container. Streams are organized inside the container directories.

The following represents the default directory structure of Origin Server:

```
/conf
 /http
  /origin.xml
 /containers
  /user A
   /container.xml
    /streams
     /channel 1
      /stream.xml
     /channel 2
```

```
      /stream.xml
  /user B
   /streams
    /channel 1
     /stream.xml
```

To support multi-tenancy, Origin Server configurations are managed by the following files:

• origin.xml

  Manages the server-level configurations.

• stream.xml

  Manages stream-level configuration within a container.

• container.xml

  Manages the container-level configurations.

Starting Origin Server

1. Unzip the distribution files.
2. Move to the `httporigin` directory.
3. On Linux, execute the following command to start the Origin Server:

   ```
   origin_start.sh
   ```

   Note: *Start Origin Server as a daemon process for the server to continue running even after the terminal is closed or the user is logged out.*

   ```
   nohup sh origin_start.sh &
   ```

Important configuration files

Origin.xml

Used to set the default port and default webroot of Origin Server. By default, the HTTP port is 8090 and the webroot is `httporigin/webroot`. This file is available in the `/HttpOrigin/httporigin/conf/http` directory.

```
<Config>

 <!--
The HTTP server listens for incoming requests on this port.
-->
 <Port>8090</Port>
 <!--
 The root folder under which content is stored.
 -->
 <Webroot>./webroot</Webroot>
 <-- This is maximum content size (in bytes) for HTTP PUT requests. This should be set to at
least the maximum expected fragment size.
 <MaxContentLength>4096000</MaxContentLength>


 <!-- Response code to be returned by Origin Server for missing fragments.
 If not configured, Origin Server will return 404. Custom response code
 for missing fragments can be used to implement failover at the caching layer.-->

 <MissingFragmentResponseCode>503</MissingFragmentResponseCode>
 <!-- Specifies TTL values in integral seconds for various file types.
```

```
 When TTL value is set for a file type, then following headers are set in Http Response of
corresponding file type:
       1) max-age header - set to TTL value
       2) Date header - set to response date
       3) Expires header - set to response date + TTL

 The file types supported are F4F, F4M, Bootstrap, CrossDomain, M3U8, HLSSegment, DRMMetadata,
 Timeline, DashSegment, DashManifest and DashManifestPart.
 -->
 <Headers>
 <TTL>
   <F4M>0</F4M>
   <Bootstrap>0</Bootstrap>
   <M3U8>0</M3U8>
   <Timeline>0</Timeline>
   <DashManifest>0</DashManifest>
   <DashManifestPart>0</DashManifestPart>
   <DRMMetadata>60</DRMMetadata>
   <F4F>60</F4F>
   <HLSSegment>60</HLSSegment>
 </TTL>
 </Headers>
</Config>
```

Stream.xml

Used to configure security options and content protection of a specific channel. This file is located in the `<httporigin-install-dir>/conf/containers/_default_/streams/_default_` directory.

The stream.xml file also contains the following configuration parameters:

Configuration	Details
//Config/Auth/DeleteSecurityToken/Key	The AES-128 bit key to be used for decrypting the DELETE token. This value shall be a string of 32 hexadecimal digits.
//Config/Auth/DeleteSecurityToken/Timeout	Optional 'Timeout' for DELETE Token values in seconds. Default value is 600 seconds (10 minutes).
//Config/CueMode	The format of cue info to be added in the JIT generated m3u8. The allowed values for this configuration option are: PT_1_0, DPISimple, and DPIScte35. The default value is DPIScte35
//Config/FaultTolerance/MultiServerManifestSyncEnabled	Setting it to true enables inclusion of the fragments in the JIT created M3U8 that are missing locally but may be present on any other origin server in the setup.
//Config/FaultTolerance/MaxLagFromLiveEdge	Maximum time (in seconds) by which JIT generated M3U8 can lag from the live edge in order to wait for a gap entry that can be filled in by another HDS source (packager). This configuration option is disabled if MultiServerManifestSyncEnabled is false.
//Config/FaultTolerance/ConstantLagFromLiveEdge	Time (in seconds) by which JIT created M3U8 will always lag from the live edge of source HDS. If there is gap, then

	there can be additional lag at most the value of MaxLagFromLiveEdge.

```
<Config>

<!-- The duration of the content window (in decimal fractions of hours).
Fragments will be removed as they roll out of the window. A value of 0.0
disables disk management. -->

<DiskManagementDuration>3.0</DiskManagementDuration>

<!-- (Optional) Auth security applied to processing inbound PUT/POST/DELETE
requests. -->

<Auth>

<!-- If the 'SecurityToken' element is present, then an auth test is applied
to all inbound PUT/POST requests that will check for a valid X-Adobe-HDS-Token
or X-Adobe-HTTP-Token request header with a valid token value. Requests not
containing this header or passing an invalid or timed-out token value receive
an error response of 401 Not Authorized. If the <SecurityToken> element is not
present, this check is not performed. -->

<!-- <SecurityToken> If <SecurityToken> element is enabled, it requires a valid
AES-128 key to use for Token value encryption/decryption, and only Packager(s)
configures with matching key can write to this Origin.
<Key>4ff4756ed68239d34d482dbc88819abc</Key> -->

<!-- Optional 'Timeout' for Token values in seconds. Default value is 600
seconds (10 minutes). If enabled, this value needs to be applied consistently here
as well as at the Packager. -->

<!-- <Timeout>600</Timeout> -->
<!-- </SecurityToken> -->
<!-- If the 'DeleteSecurityToken' element is present, then an auth test is applied
to all inbound DELETE requests that will check for a valid X-Adobe-HDS-Token or
X-Adobe-HTTP-Token request header with a valid token value. Requests not
containing this header or passing an invalid or timed-out token value receive an
error response of 401 Not Authorized. If the <DeleteSecurityToken> element is not
present, this check is not performed. -->

<!-- <DeleteSecurityToken> If <DeleteSecurityToken> element is enabled, it requires
a valid AES-128 key to use for Token value encryption/decryption, and only clients
configured with matching key can delete from this Origin.
<Key>4ff4756ed68239d34d482dbc88819abc</Key> -->

<!-- Optional 'Timeout' for Token values in seconds. Default value is 600 seconds
(10 minutes). If enabled, this value needs to be applied consistently here as well
as at the Packager. -->

<!-- <Timeout>600</Timeout> -->
<!-- </DeleteSecurityToken> -->

</Auth>

<HttpConfig>

<!-- Uncomment following to override Server TTL settings for this stream.
<Headers>
 <TTL>
  <F4M>0</F4M>
  <Bootstrap>0</Bootstrap>
  <M3U8>0</M3U8>
  <Timeline>0</Timeline>
  <DashManifest>0</DashManifest>
  <DashManifestPart>0</DashManifestPart>
 </TTL>
```

```
</Headers> -->

</HttpConfig>

<!--
Uncomment the following to enable Fault tolerance support for this stream.
These options are useful for fault tolerance in workflow involving Just-In-Time
HLS generation from HDS artifacts, when redundant servers are deployed for this
stream. When <FaultTolerance> tag exists, then the contained tags are enabled with
the default values listed below.

MultiServerManifestSyncEnabled: Set it to true if this stream will be served from
multiple origins using 503-failover-proxy-setup (as described in
http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/
adobe-media-server/pdfs/rock-solid-live-streaming.pdf)

It turns on inclusion of entries of fragments in the JIT created M3U8 that might
be missing on the local origin server, but may be present on any other origin
server. When this is set, HLS manifest is updated using the HDS manifest pushed
from the packager and can recover the fragments missed while origin was
down.

MaxLagFromLiveEdge: Live edge of JIT created HLS stream will lag that of
source HDS by at most the amount (in integral seconds) specified by this option.
It is assumed to be 0 if<MultiServerManifestSyncEnabled> is not enabled. In
practice, JIT HLS will lag HDS only when some fragment is found missing, otherwise
it will be near the live edge.If the packagers (sources of HDS fragments) are not
in sync exactly, then this option allows recovery from the other packager if the
leading packager drops or is unable to push a fragment, and the slower packager
manages to send the missing fragment within the configured threshold lag.

ConstantLagFromLiveEdge: It shifts the live edge to delay the JIT HLS stream. It
should be set to a value (in seconds) by which the fragments can arrive out of
order, for example: when Packager is configured to push the fragments in parallel
as soon as they are created rather than serially (waiting for previous one to be
sent, before pushing next fragment).

MaxLagFromLiveEdge will apply on top of the delay set in ConstantLagFromLiveEdge,
so effectively JIT HLS stream is constantly delayed by ConstantLagFromLiveEdge and
can incur a maximum delay of(ConstantLagFromLiveEdge + MaxLagFromLiveEdge) on
occurrence of gap.
-->

<!--
<FaultTolerance>
 <MultiServerManifestSyncEnabled> true </MultiServerManifestSyncEnabled>
 <MaxLagFromLiveEdge> 10 </MaxLagFromLiveEdge>
 <ConstantLagFromLiveEdge> 0 </ConstantLagFromLiveEdge>
</FaultTolerance>
-->

<!-- The format of cue info to be added in the JIT generated f4m/m3u8. -->
<CueMode>DPIScte35</CueMode></Config>
```

Server.xml

Contains the default content protection settings to enable AAXS DRM for HDS or HLS streams. This file is located in the `httporigin/conf` directory.

Container.xml

Each container directory must have a container.xml file in order for it to be recognized as a stream container.

You can specify the security token and TTL settings in the container.xml file. These settings apply to set level manifests published to container. You can specify separate security tokens for PUT/POST and DELETE requests.

You can also enable security token for PUT/POST and DELETE requests independent of each other. For example, you can enable a security token only for DELETE requests and disable it for PUT/POST requests or vice versa.

Note:

The container settings are not inherited by the stream/vod module. To apply security or TTL settings on a stream/vod module, you must apply the corresponding configuration in the stream.xml/vod.xml file.

The following are sample contents of a container.xml file:

```
<Config>

<!-- Note that all the config specified here is applicable only on the container path and these

settings are NOT inherited by the stream/vod modules under this container. To apply security
or
TTL settings on a stream/vod module, corresponding config must be applied in stream.xml/vod.xml
-->

<!-- (Optional) Auth security applied to processing inbound PUT/DELETE requests on this
container.
<Auth>
-->

<!-- If the 'SecurityToken' element is present, then an auth test is applied to all inbound
POST
requests that will check for a valid X-Adobe-HDS-Token or X-Adobe-HTTP-Token request header
with
a valid token value. Requests not containing this header or passing an invalid or timed-out
token
value receive an error response of 401 Not Authorized. If the <SecurityToken> element is not
present,
this check is not performed.
<SecurityToken>
-->

<!-- If <SecurityToken> element is enabled, it requires a valid AES-128 key to use for Token
value
encryption/decryption, and only Packager(s) configures with matching key can write to this
Origin.
-->

<!-- <Key>4ff4756ed68239d34d482dbc88819abc</Key> -->
<!-- Optional 'Timeout' for Token values in seconds. Default value is 600 seconds (10 minutes).

If enabled, this value needs to be applied consistently here as well as at the Packager. -->

<!-- <Timeout>600</Timeout> </SecurityToken>-->

<!--
If the 'DeleteSecurityToken' element is present, then an auth test is applied to all inbound
DELETE requests that will check for a valid X-Adobe-HDS-Token or X-Adobe-HTTP-Token request
header
with a valid token value.

Requests not containing this header or passing an invalid or timed-out token value receive an
 error
response of 401 Not Authorized. If the <DeleteSecurityToken> element is not present, this check
 is not
performed.
-->

<!--
<DeleteSecurityToken>

If <DeleteSecurityToken> element is enabled, it requires a valid AES-128 key to use for Token
```

```
 value
encryption/decryption, and only clients configured with matching key can delete from this
Origin.

<Key>4ff4756ed68239d34d482dbc88819abc</Key>
-->

<!-- Optional 'Timeout' for Token values in seconds. Default value is 600 seconds (10 minutes).
 If enabled,
this value needs to be applied consistently here as well as at the Packager. -->

<!-- <Timeout>600</Timeout> -->
<!-- </DeleteSecurityToken> -->
<!--</Auth> -->

<HttpConfig>

<!-- Uncomment following to override Server TTL settings for the GET requests made on this
container path.
<Headers>
 <TTL>
  <F4M>0</F4M>
  <Bootstrap>0</Bootstrap>
  <M3U8>0</M3U8>
  <Timeline>0</Timeline>
  <DashManifest>0</DashManifest>
  <DashManifestPart>0</DashManifestPart>
 </TTL>
</Headers>
-->

</HttpConfig>
</Config>
```

Configuring Origin Server

Changing default Origin Server settings

1. Extract the Origin Server package.
2. Open the `httporigin/conf/http/origin.xml` file.
3. Set the following properties:
   - Webroot
   - Port
   - The folder where the log files are to be written
   - Maximum content length
   - `MissingFragmentResponseCode`:

     Response code to be returned by Origin Server for missing fragments. If this property is not configured, Origin Server returns 404 for missing fragments. Use custom response code for missing fragments to implement failover at the caching layer.
   - TTL Header values: Specifies TTL values in integral seconds for various file types.

     If TTL values are not set up for a file type, the following headers are set in the file type's http response:

     - max-age header, set to TTL value
     - Date header, set to response date
     - Expires header, set to response date + TTL

The following file types are supported:

- F4F
- F4M
- Bootstrap
- CrossDomain
- M3U8
- HLSSegment
- DRMMetadata
- Timeline

Configuring security options for a stream

You can configure Origin Server to allow only a Packager with matching token key, module name, and stream name to write to a container.

Note: *Token key prevents unauthorized access to data. It is a server-specified data that is generated when a 401 response is made.*

The stream or module hierarchy at Packager need not necessarily match that of Origin Server. The stream configuration at Origin Server and Packager are completely independent of each other. You can configure one stream at Packager to publish to two different streams at Origin Server.

1. Open `stream.xml`, located in the respective channel directory.
2. Configure the following properties:

Option	Description
Security Token	If the <SecurityToken> element is present, an auth test is applied to all inbound POST requests. The test verifies if there is a valid X-Adobe-HTTP-Token request header with a valid token value. Requests that don't contain this header or pass an invalid or timed-out token value receive an error response (401 Not Authorized). If the <SecurityToken> element is absent, this check is not performed. The Security Token is not enabled by default for modules. The token is expected only if the Auth/SecurityToken tag is present in the applicable module/origin config file. If Auth/SecurityToken is present, a valid AES 128 Key must be provided in the Key tag. No default key is used for token generation. If the Auth/SecurityToken tag is present and a Key is not provided, a ModuleConfigException messare is raised.
Key	If this Key is enabled, a valid AES-128 key is required to use for Token value encryption or decryption. Only Packager(s) configured with the matching key can write to this Origin. If not enabled, a default system key matching the Packager's default is used. Following is the default key: `<key>4ff4756ed68239d34d482dbc88819abc</key>`

Option	Description
	The packager need not specify the default key.
Timeout	This token is optional. Default value is 600 seconds (10 minutes). If enabled, this value needs to be applied consistently in stream.xml and in the Packager.

3. Save the file and restart the server.

Supporting legacy players

1. Open `stream.xml` located in the respective channel directory.
2. Uncomment the following line of code:
   ```
   <WriteExternalBootstrap>true</WriteExternalBootstrap>
   ```

Managing disk space

To allow 24/7 live streaming, Origin Server implements a rolling disk space management scheme.

This can be configured in the *origin.xml* file. The *<DiskManagementDuration>* property in the *origin.xml* file specifies the duration of the content in hours. Fragments that fall within this duration are kept. If the fragment falls out of the duration, the bootstrap file is updated to remove it from the fragment run table. Then, the actual fragment is deleted from the disk.

1. Open the stream.xml in the respective channel directory.
2. Update the <DiskManagementDuration> element.
   The default value is 3 hours. To disable the property, set it to 0.

Fault Tolerance for HDS and just-in-time HLS

Fault Tolerance for HDS is implemented by deploying origin servers, packagers, and encoders redundantly. This deployment optimizes the HDS workflow because the client can obtain the missing fragments in the bootstrap using client-side Best Effort Fetch (BEF) and server-side 503 failover processes, described in *OSMF Best Effort Fetch Functional Spec* and *Rock Solid Live Streaming*.

For HLS workflow, client-side BEF is not readily available. In the HLS workflow, the URL of a segment, missing in m3u8 (HLS manifest), cannot be predicted easily. This is because HLS segments must be aligned with the cue points. Therefore, a functionality equivalent to BEF must be introduced in the server system. BEF-type functionality is achieved by including fragments available in the whole server system (not just at the local origin server) in the HLS manifest. See *Deployment* guide to know more about the JIT fault tolerance implementation and configuration options for HDS and HLS workflow.

Using default content protection

Origin Server applies the default certificates from the root configuration file to enable HDS and HLS content protection using default certificates on any stream.

Enable the ContentProtection tag in the config.xml file.

The root configuration file (server.xml) is located in the httporigin/conf directory. Following is a sample root configuration file:

```
<Config>
    <ContentProtection>
        <FAXS4>
            <LicenseServerURL>http://primetime.aaxs.adobe.com </LicenseServerURL>

<LicenseServerCertificate>creds/static/phds_license_server.der</LicenseServerCertificate>

<LicenseServerCredential>creds/static/phds_license_server.pfx</LicenseServerCredential>
            <LicenseServerCredentialPassword>xxxxxxxxxxx</LicenseServerCredentialPassword>
          <PackagerCredential>creds/static/phds_production_packager.pfx</PackagerCredential>

            <PackagerCredentialPassword>xxxxxxxxxxx</PackagerCredentialPassword>

<TransportCertificate>creds/static/phds_production_transport.der</TransportCertificate>
            <CommonKey>creds/common-key.bin</CommonKey>
            <PolicyFile>creds/static/phds_policy.pol</PolicyFile>
            <RecipientCertificates>creds/sd</RecipientCertificates>
        </FAXS4>
    </ContentProtection>
</Config>
```

Configuration	Details
ContentID	Content ID used to generate the CEK. It's an optional parameter. If value is not provided, stream ID is used for live case. For VOD, relative asset path is used.
KeyRotation	Specifies whether the key rotation feature is enabled. The values can be true or false. It's an optional parameter. Default value is false.
KeyRotationInterval	Time interval in seconds during which the same rotation key is used to encrypt content samples. It's an optional parameter. Default value is 900 seconds.
LicenseServerUrl	The license server's URL.
KeyServerURL	The key server's URL.
LicenseServerCertificate	Path of the file containing license server certificate.
RecipientCertificates	Path to the directory containing the shared domain certificates. It is an optional parameter. If RecipientCertificates path is not provided for EmbedLicense protection scheme in stream.xml/vod.xml, the path is taken from the root configuration.
LicenseServerCredential	Path of the . pfx file containing license server credential.
LicenseServerCredentialPassword	Password to encrypt license server credential.
PackagerCredential	Path of the . pfx file containing packager credential.
PackagerCredentialPassword	Password for protecting packager credential.
TransportCertificate	Path of the file containing transport certificate.
PolicyFile	Path of the file containing policy.

CommonKey	File containing common key of size at least 16-bytes. Common key is used together with content ID to derive CEK.
WhitelistFolder	Path of folder containing whitelist app information. This is only used with PHDS/PHLS. It's an optional parameter.

Supported content protection features

- License chaining: HLS DRM using FAXS protection scheme makes use of Enhanced License Chaining similar to HDS DRM. HLS Packager or Origin Server generates leaf license containing CEK- encrypted using a root key and embed it into the DRM metadata. FAXS iOS SDK requests root license from License Server and then extract CEK from leaf license using root license. Policy file used needs to specify the root policy for FAXS protection scheme. For PHDS/PHLS/chained license, the cache headers of JIT DRM response from Origin are set as per the license validity date. If TTL values are also specified in the configuration, the license end date overrides the corresponding values in the configuration. For PHDS/PHLS/chained license, the cache headers of JIT DRM response from Origin is set as per the license validity date. If TTL values are also specified in the configuration, the license end date overrides the corresponding values in the configuration.

  Note:

  Primetime Origin does not create an expired leaf license for chained license policy. Leaf license properties, such as validity, start date, and end date are determined from the policy. To create an expired leaf license to enforce the license validity from root license, specify the end data in the policy.

- Key rotation: If Key Rotation is enabled, instead of encrypting content directly with the Content Encryption Key (CEK), the Rotation Key (RK) is used to encrypt the content. The Rotation Key changes in the specified key rotation interval. Each Rotation Key is encrypted with CEK.
  - HLS Key rotation: If key rotation is enabled, FAXS API provides the encrypted rotation key used to encrypt ts fragments. This encrypted rotation key is specified as value for the EncryptedRK parameter in the key URI in m3u8.
  - To support random access, the encrypted rotation key is provided at the beginning of each encrypted audio or video packet. To decrypt the packet, the player uses CEK to decrypt the rotation key. Then uses the rotation key to decrypt the content.
- PHDS/PHLS: For PHDS or PHLS, License Server is not required, because the license is embedded in the DRM metadata. To enable PHDS or PHLS, policy file specified should have LicenseBindingType set for embedded license. One or more recipient certificates of the type SharedDomain can be configured at the time of generation of DRM metadata. The Recipient key certificates (private key associated with the Shared Domain certificates) are included in the FAXS client.

  Path of directory containing Shared domain certificates will be configured in ContentProtection/RecipientCertificates. The set of shared domain certificates may change over time in response to security breaches or other reasons. In such events, new Shared Domain Certificates will be released. Upon notification from Adobe or a security alert, customers need to retrieve new Shared Domain certificates from a secured Adobe website, which would host those updates and update them in the directory specified in ContentProtection/RecipientCertificates.
- App whitelisting: App whitelisting can be used for PHDS/PHLS protection scheme. App whitelisting restricts playback of encrypted content only to the specified apps/swfs. app Information (Swf Hash/iOS App Information) is included in the embedded license. FAXS client ensures that the app information provided in the embedded license matches

that of the app used for playback. If the user wants to enable app whitelisting for PHDS/PHLS, they can specify the WhitelistFolder tag in the configuration file.

Configuration examples

m3u8 containing DRM information (for local key delivery mode)

```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-FAXS-CM:URI="livestream.drm"
#EXT-X-KEY:METHOD=AES-128, URI="faxs://faxs.adobe.com",IV=0x15161718191A1B1C1D1E1F2021222324
#EXT INF:10, http://orgserver/live/sample/livestream23585170.ts
#EXTINF:10, http://orgserver/live/sample/livestream23589174.ts
```

m3u8 containing DRM information (for remote key delivery mode)

```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-FAXS-CM:URI="livestream.drm"
#EXT-X-KEY:METHOD=AES-128,
 URI="https://keyserver.com",
 IV=0x15161718191A1B1C1D1E1F2021222324
#EXTINF:10,http://orgserver/live/sample/livestream23585170.ts
#EXTINF:10, http://orgserver/live/sample/livestream23589174.ts
#EXTINF:10, http://orgserver/live/sample/livestream23593178.ts
```

m3u8 containing key rotation information

```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-FAXS-CM:URI="livestream.drm"
#EXT-X-KEY:METHOD=AES-128,
 URI="faxs://faxs.adobe.com?EncryptedRK=0x00112233445566778899aabbccddeeff",
 IV=0x15161718191A1B1C1D1E1F2021222324#EXTINF:10,
 http://orgserver/live/sample/livestream23585170.ts
#EXTINF:10, http://orgserver/live/sample/livestream23595170.ts
#EXTINF:10, http://orgserver/live/sample/livestream23605170.ts
#EXT-X-KEY:METHOD=AES-128,
 URI="faxs://faxs.adobe.com?EncryptedRK=0x0068899a112233bb677ccddee4455aff",
 IV=0x15161718191A1B1C1D1E1F2021222324#EXTINF:10,
 http://orgserver/live/sample/livestream23615170.ts
#EXTINF:10, http://orgserver/live/sample/livestream23625170.ts
#EXTINF:10, http://orgserver/live/sample/livestream23635170.ts
```

JIT trick play support

To enable trick play, no additional configuration changes are required in the Origin Server. Trick play assets can be created while accessing them using the URL that contains the kfonly or HDS-KFOnly tags.

When a file that contains the kfonly or HDS-KFOnly tags in the URL is accessed through the Origin server, it's is created from the asset located at the same URL if it doesn't exist. In this case:

- The F4M or Bootstrap at the location kfonly/ contains the same content as the parent asset.
- The HDS fragment at the location kfonly/ URL is created by copying the source fragment. However, only the first AVCC tc message and the first keyframe TCM message are retained in the mdat box. All other TCMessages are deleted. If source fragment contains moof box, it is not included in the output fragment.

For example, if the file at the URL http://localhost:8090/_default_/_default_/kfonly/livestreamSeg1-Frag39679is accessed and the file doesn't exist on the disk, then a keyframe-only HDS fragment is generated just-in-time from the source fragment at the URL http://localhost:8090/_default_/_default_/livestreamSeg1-Frag39679.

The following is a sample set level F4M that lists a JIT derivable trick play stream:

```
http://ns.adobe.com/f4m/2.0"
<manifest xmlns=">
 <baseURL>http://localhost:8090/_default_/_default_/</baseURL>
 <dvrInfo windowDuration="1800" />
 <media href="livestream.f4m" bitrate="1500" />
 <media href="kfonly/livestream.f4m" type="video-keyframe-only" bitrate="1500" />
</manifest>
```

For HLS, trick play assets can be JIT created on Primetime Origin Server 1.2 and accessed using a URL containing kfonly tags. For a published live/vod stream, the corresponding trick play stream will be available at:

Original URL	JIT Trick Play URL
http://host:port/container/module/rel_path/streamName.m3u8	http://host:port/container/module/rel_path/kfonly/streamName.m3u8

The trick play m3u8 is generated as per the requirements mentioned in *section 3.4.12* of HLS Specifications.

HLS Trick Play requires HLS version 4 or higher. The value of the EXT-X-VERSIONtag in all JIT Trick Play HLS streams is 4 or higher (even if the corresponding original stream is at a lower version).

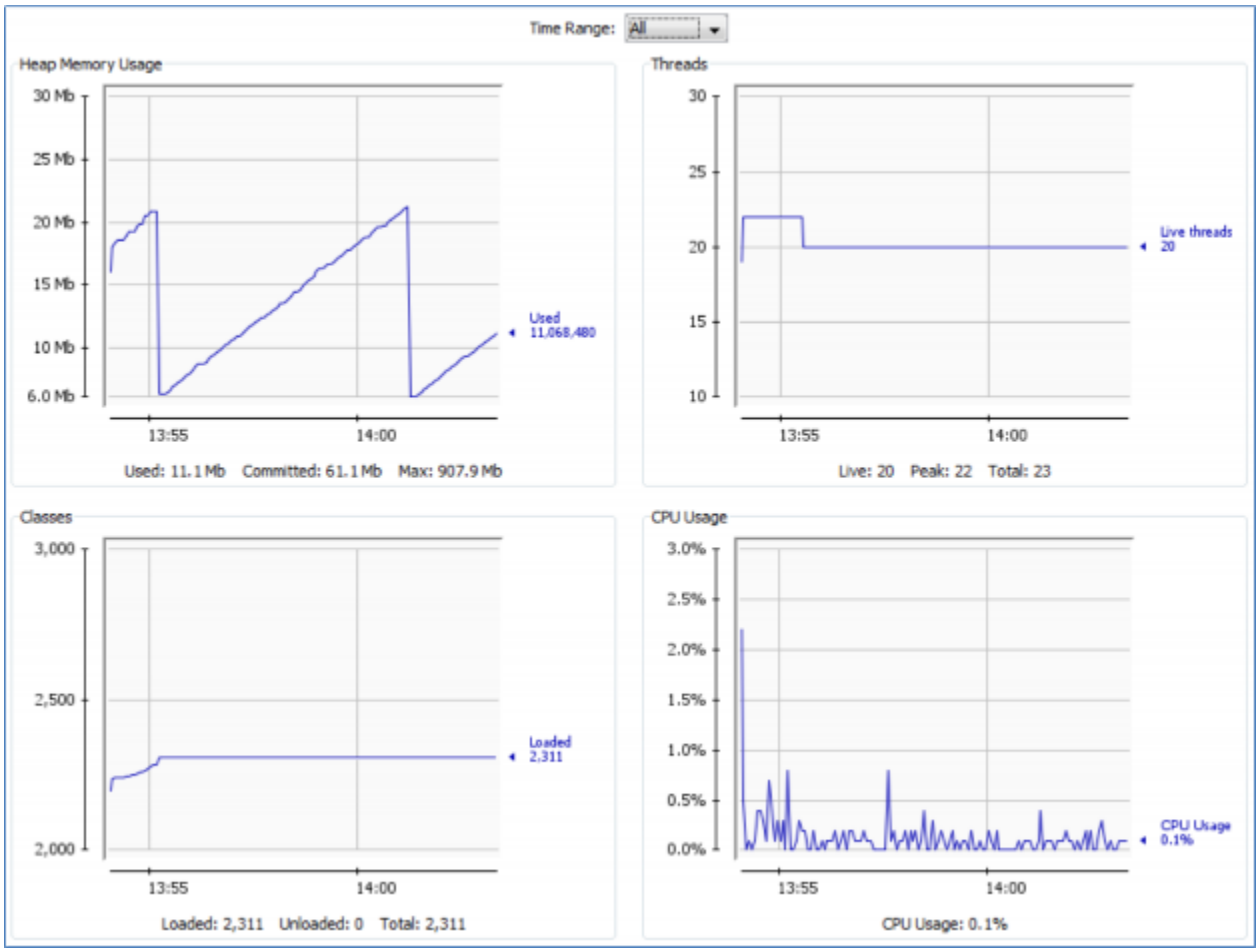
For the above example, it is possible that http://host:port/container/module/rel_path/streamName.m3u8 has #EXT-X-VERSION:3 whereas http://host:port/container/module/rel_path/kfonly/streamName.m3u8 can have #EXT-X-VERSION:4

For HLS, JIT trick play is available for both the HLS streams directly published to Origin and those that are JIT generated at Origin.

Note: *For this feature to work, the segment URIs inside the m3u8 should be relative URIs. If m3u8 contains absolute segment URI, JIT Trick Play doesn't work.*

JMX administrator console

Use the JMX administrator console to monitor statistics and perform various operations.



Loading newly added containers and streams

The JMX management interface can be accessed by jConsole, available in the JDK directory.

1. Add new container or stream in the conf folder.
2. Run jconsole from the command line.
3. In the New Connection dialog box, select **origin.jar.com.adobe.fms.OriginServer** from Local Process.
4. Click **Connect**.
5. Select **MBeans**.
6. Refresh the newly added container or stream:
   a) Go to com.adobe.fms > Operations.
   b) Click **Refresh**.
7. Dynamically load a specific module ID:
   a) Go to com.adobe.fms > StreamContainer.
   b) Click the Refresh button associated with the module ID.

Using JMX operations

1. In the JMX console, go to StreamContainer > Lifecycle > Operations.
2. Perform the following operations:

Option	Description
Refresh()	Loads any new descendant modules that have been added to the system after this module was initialized. This module and any descendants that are already loaded and initialized are not affected apart from new child modules being optionally wired to their parent module. **Valid operation level** : Server **Parameters** : NIL **Returns** :StatusCode:Messageformat
Shutdown()	Shuts down the module and all descendant modules. **Valid operation level**: Server **Parameters** : NIL **Returns** :StatusCode:Messageformat
Reset()	This operation removes the binding for the modules' request handler, so that new requests are not serviced during a reset. Then the webroot and moduleRequestUriPrefix are resolved to the modules' respective directory and all contents are recursively deleted. A new Stream request handler will be created using the same configuration settings. Note that the new settings require a shutdown or a refresh. Stats for the Stream module will not be reset until the module is shutdown or the server is restarted. Invoking reset on a Stream Container module resets all the child modules. **Valid operation level**: stream container and stream module **Parameters**: NA **Returns**: StatusCode:Messageformat
setConfig (String xmlConfig)	This should be invoked on an Origin Server stream. It updates the module's xml configuration on disk with the specified xml configuration. This operation does not reload the module. Configuration changes are only reflected when module is reset using the reset() operation. Before replacing the configuration file on the disk, the given xml is validated to ensure that it is a well-formed xml. ***Note:*** *Attribute validation is not performed on the XML file.*

Option	Description
	Valid operation level: Stream module, VOD module, and Stream Container module **Parameters**: XML configuration file **Returns**: StatusCode:Messageformat This should be invoked on an Origin Server stream or VOD module. It updates the module's xml configuration on disk with the specified xml configuration. This operation does not reload the module. Configuration changes are only reflected when module is reset using the `reset()` operation. Before replacing the configuration file on the disk, the given xml is validated to ensure that it is a well formed xml. **Parameters**: XML configuration file **Returns**: IModule.StatusCode as a string **Message on successful operation**: String "success.completed" **Errors**: • String "`error.failed`" for an unsuccessful operation. • String "`error.not_implemented`" if the operation is invoked on a module that does not support it.
getConfig(String xmlConfig)	It should be invoked on an Origin Server Stream or VOD module. This reads the existing xml configuration for the given module from the disk and returns the configuration as a string. It does not affect the current state or configuration of the module. **Valid operation level**: VOD module, stream container module, and stream module **Parameters**: None **Returns**: The existing XML configuration. **Message on successful operation**: None **Errors**: • String "`error.failed`" for an unsuccessful operation. • String "`error.not_implemented`" if the operation is invoked on a module that does not support it.
String createChildModule(String moduleName, String moduleType, String moduleConfig)	This should be invoked on a StreamContainerModule of Origin Server. It is used to create a new stream or VOD module under the given StreamContainerModule. This operation verifies the existing modules under the StreamContainerModule to check if there is already another module by the given name. This operation also checks if the configuration xml is well formed. If both

Option	Description
	these validations succeed, then it creates a new directory for the new module and then copies the configuration xml to the new module. This operation does not instantiate the new module or binds it to the parent module. It creates the required xml configuration on the disk. To instantiate and bind the module to the parent module, the `refresh()` operation must be invoked on the parent module. **Parameters**: • Module name: name of the child module. • Module type: Type of the module. It can be Stream or vod. • Module configuration file. **Returns**: StatusCode:Messageformat **Valid operation level**: StreamContainerModule
public String removeChildModule(String moduleName, String moduleType)	This operation can be invoked on a Stream Container module in Origin Server to remove the `conf` folder of a Stream or VOD Module with given Module ID. This operation does not remove a running module. To remove `conf` for a child module, the child must first be stopped using the shutdown() operation. In case this operation is invoked to remove a running module, error message is returned. Also, if the configuration for the module to be removed does not exist on the disk, it returns an error string. This operation only removes the config folder for the specified folder and does not perform any other cleanup. The `Rest()` method provided by the Module's LivecyclyManager MBean must be used for clean-up purposes. **Valid operation level**: StreamContainerModule **Parameters**: • Module name: name of the child module. • Module type: Type of the module. It can be Stream or vod. • Module configuration file. **Returns**: StatusCode:Messageformat

Status code:Message format

The JMX methods return a string that is a combination of a numeric status code and a readable message. The message is optional and it can be empty. The status code is terminated by a colon. The code represents the execution status.

The following table displays various status codes and their descriptions:

Status Code	Description
0	Success
1	Unknown error
2	Method not implemented
3	Cannot create a module by the given name, because a module by that name already exists.
4	Cannot delete or update the module, because it does not exist.
5	Cannot delete module, because it is running.
6	Operation failed due to invalid configuration.
7	Invalid module name provided as the parameter.
8	Some child modules failed to load when Refresh is called on a container.

JMX statistics

JMX Stats MBean is also exposed for Stream Container Module, in addition to the Stream Module and VOD Module. JMX Stats MBean expose the following attributes:

Name	Type	Description	Example
get_requests	long	The number of GET requests to the module	756
put_requests	long	The number of PUT/POST requests to the module	231
delete_requests	long	The number of DELETE requests to the module	7

get_errors	String	It's a string representation of the number of error response(4xx/5xx) for HTTP GET requests on the given module. The format for the String is:
		`errorCode1 : counter1 ;`
		`errorCode2 : counter 2 ;`
			500 : 10 ; 503 : 178 ;

put_errors	String	It's a string representation of the number of error response(4xx/5xx) for HTTP PUT/POST	409 : 72 ; 500 : 283 ;

Name	Type	Description	Example

		requests on the given module. The format for the String is:
		`errorCode1 : counter1 ;`
		`errorCode2 : counter 2 ;`	

delete_errors	String	It's a string representation of the number of error response(4xx/5xx) for HTTP DELETE requests on the given module. The format for the String is:
		`errorCode1 : counter1 ;`
		`errorCode2 : counter 2 ;`
			409 : 72 ; 500 : 283 ;

Log information

`origin.log` and `access.log` are the two log files generated for Origin Server.

`origin.log` contains the logs related to HTTP Origin. `origin.log` is available in the `\httporigin\logs` folder.

`access.log` rotates according to the rotation policy. `access.log` contains one line per HTTP request made to Origin Server. `access.log` is available in the `\httporigin\logs\http` folder.

The information captured for ever HTTP request is as follows:

• LocalTime on the server
• HTTP verb
• Source IP
• HTTP error code
• Requested URL
• Request length
• Response length
• Time taken to serve out the request in milliseconds

Logging namespace

The following are the logging namespace:

Logging namespace	Description
com.adobe.fms	The root namespace for all Origin logging. Output is set to *./logs/origin.0.log* by default.
com.adobe.fms.http	Logging for the built-in http server. It logs all requests and responses. By default, output is set to *./logs/http/access.0.log.*
com.adobe.fms.StreamContainer.streamContainerName	*streamContainerName* is a variable based on configured containers. The container logs and child stream logs may attach a FileHandler for diagnostic purposes in the

	logging.properties file. By default, there is no filehandler setup.
com.adobe.fms.StreamContainer.streamContainerName.Stream.streamName	*streamContainerName* and *streamName* are variable based on the configured container and stream. A specific stream's logs may attach a FileHandler for diagnostic purposes in the *logging.properties* file. By default, there is no filehandler setup.

Note: *Logging output from the stream level is prefixed with containerName,streamName. Logging output from the container level is prefixed with the containerName.*

Setting the log levels using JMX console

The logger level defines the logging output. By default, INFO is set as the log level. All the Java-related logs are captured in the origin.log file. You can change the log level using the JMX admin console.

Following are the various logger level that you can set using java.util.logging class:

- Level .SEVERE (highest value)
- Level .WARNING
- Level .INFO
- Level .CONFIG
- Level .FINE
- Level .FINER
- Level .FINEST (lowest value)

For more information on logger levels, see *java.util.logging* class.

1. Query using JMX to the LoggerNames attribute found on the java.util.logging mbean. Identify the stream for which you want to modify the log level.
   For example, com.adobe.fms.StreamContainer.default.Stream.default.
2. Call setLoggerLevel on the java.util.logging mbean with the logger name identified, and specify which log level you want to set.
3. Call setLoggerLevel as shown below to set the logger level to FINEST :
   setLoggerLevel("com.adobe.fms.StreamContainer.default.Stream.default", Level.FINEST).

   Each individual stream should log to its own particular log file. The log file rolls over based upon the settings of the FileHandler in logging.properties.

Setting the log levels using logging.properties file

Log levels may also be manipulated through the use of the *logging.properties* file.

The following example shows how to configure a FileHandler for the container, "*com.adobe.fms.StreamContainer.foo*" in the *logging.properties* file.

```
com.adobe.fms.module.FileHandler1.pattern = ./logs/debugContainer.%g.log
com.adobe.fms.module.FileHandler1.level = FINEST
com.adobe.fms.module.FileHandler1.formatter = java.util.logging.SimpleFormatter
com.adobe.fms.module.FileHandler1.append = true
com.adobe.fms.module.FileHandler1.limit = 524288000
com.adobe.fms.module.FileHandler1.count = 4
```

```
com.adobe.fms.StreamContainer.foo.handlers = com.adobe.fms.module.FileHandler1
com.adobe.fms.StreamContainer.foo.level = FINEST
```

The following example shows how to configure a FileHandler for the stream "*com.adobe.fms.StreamContainer.foo.Stream.Stream1*" in the*logging.properties* file.

```
com.adobe.fms.module.FileHandler2.pattern = ./logs/debugStream.%g.log
com.adobe.fms.module.FileHandler2.level = FINEST
com.adobe.fms.module.FileHandler2.formatter = java.util.logging.SimpleFormatter
com.adobe.fms.module.FileHandler2.append = true
com.adobe.fms.module.FileHandler2.limit = 524288000
com.adobe.fms.module.FileHandler2.count = 4
com.adobe.fms.StreamContainer.foo.Stream.stream1.handlers = com.adobe.fms.module.FileHandler2
com.adobe.fms.StreamContainer.foo.Stream.stream1.level = INFO
```

Note: *After updating the logging.properties file, restart the process.*

Adobe file handlers

By default, there are 10 diagnostic FileHandlers. They are named from *com.adobe.fms.module.FileHandler0* to *com.adobe.fms.module.FileHandler9*

If you need more FileHandlers, compile them and place them in the*classpath* of the *origin_start*script.

Note: *java.util.logging.LogManager mandates that the directory must exist when specified as a pattern. If not, an exception is thrown that it was unable to obtain a lock for the file.*

Just-in-time generation of HLS assets

Primetime Origin Server supports just-in-time (JIT) creation of HLS assets, such as ts and m3u8 from HDS assets created by Primetime Packager or Encoders.

For more information, see the *Adobe Primetime Delivery Protocol Specification* document.

For every HDS live or VOD stream published to Primetime Origin, the corresponding HLS stream is available by default through JIT-conversion.

Following are the prerequisites for HDS to HLS JIT-conversion:

- The HDS stream published to the Origin Server should follow the Adobe Primetime Delivery Server protocol.
- The published HDS stream should not be DRM protected. For DRM-protected HDS stream, the request for corresponding JIT-converted HLS stream returns a 403 response code (forbidden). To enable encryption on such streams, use JIT-encryption. Encryption configuration should be specified in the stream.xml or vod.xml and then both the HDS and corresponding JIT-converted HLS stream are JIT-encrypted.
- HDS stream should not contain external bootstrap. In the Stream configuration on Origin Server, the WriteExternalBootstrap tag shouldn't be set to true. For such a stream, the request for corresponding JIT-converted HLS stream returns a 403 response code (forbidden).
- The f4m published should contain either the fragmentDuration attribute in the bootstrapInfo element or the targetDuration element under the manifest element. These are required to determine the target duration for HLS stream.

JIT m3u8 generation

The m3u8 file for live HDS stream is created when the HDS stream is published to the Origin Server (PUSH). The m3u8 is generated incrementally and also serialized on disk to ensure the consistent sequence numbering if Origin Server is restarted.

In this case, **the DVR window of the generated live m3u8 = (DVR window of HDS/2) - Target Duration**. The target duration of VOD m3u8 is set as the duration of the longest fragment.

The target duration of the live -m3u8 is determined as follows:

- If the bootstrapInfo element of the f4m contains the fragmentDuration attribute, then the Target Duration is considered as **(2.1 * fragmentDuration)**.
- If the bootstrapInfo element does not have the fragmentDuration attribute, the targetDuration element is looked up. The targetDuration element is expected as a child element under the manifest element and should contain the target duration in fractional seconds. If this element is present, its value is used as the Target Duration.

HLS manifest tags for synchronization

Primetime Origin Server supports the following tags in the HLS manifest file for synchronization across renditions:

Tag name	Description
#EXT-X-MEDIA-TIME	The just-in-time generated M3U8 at origin server contains this custom tag for synchronization across renditions
#EXT-X-PROGRAM-DATE-TIME	This tag is generated in JIT created M3U8 using SMPTE timecodes in the HDS manifest, in the following way: • Value of attribute @smpte of <smpteTimecode> element in HDS manifest, used to derive the TIME part. • For multiple <smpteTimecode> elements whose @timestamp lies in a single fragment, only the first time-code is utilized. • If <smpteTimecode> is not already aligned with the fragment-start-timestamp, the value of smpte attribute is adjusted in a manner that the resulting time is aligned with start of the fragment where the #EXT-X-PROGRAM-DATE-TIME tag is applied. • If @timezone is not specified in <smpteTimecode>, GMT is assumed in #EXT-X-PROGRAM-DATE-TIME. • If date is not provided, the current system/wall-clock date is taken.

JIT Audio Only

Primetime Origin Server supports just-in-time audio only conversion for live and VoD streams for both HDS and HLS. The following table shows the location of corresponding JIT audio streams for any published live/VoD stream:

Original URL	JIT Audio Only URL
http://host:port/container/module/rel_path/streamName.f4m	http://host:port/container/module/rel_path/audio_only/streamName.f4m
http://host:port/container/module/rel_path/streamName.m3u8	http://host:port/container/module/rel_path/audio_only/streamName.m3u8

Access Plugin for remote mp4

The HTTP access plugin lets you modify remote mp4 HTTP requests for mp4 files that are fetched from a remote HTTP server. You can use the plugin to add or remove headers and rewrite urls. Primetime Origin Server 1.2 provides the following default implementations of the HTTP access plugin:

- com.adobe.fms.util.httpfileaccess.SimpleHttpAccess: Resolves the mp4 url against the base URL of remote http server on which mp4 assets are hosted.
- com.adobe.fms.util.httpfileaccess.S3HttpAccess: Resolves the mp4 request against the configured S3 bucket. Also, it computes the request token for accessing the mp4 and adds the required headers to the mp4 http request.

To use a plugin, specify its name and the corresponding configuration in the vod. xml file under the Config/ MP4 element.

You should also specify the fully-qualified class name of the plugin in the Config/ MP4/ HttpAccessPlugin/ Class element. You can specify additional configuration for the plugin under Config/ MP4/ HttpAccessPlugin/ Init.

The following is a sample configuration for the SimpleHttpAccess plugin:

```
<MP4>
 <!-- The duration of the output fragments in milliseconds. -->
<FragmentDuration>4000</FragmentDuration>
<!-- The TargetDuration should be greater than the maximum fragment duration in milliseconds.
 -->
<TargetDuration>6000</TargetDuration>
<HttpAccessPlugin>
<Class>com.adobe.fms.util.httpfileaccess.SimpleHttpAccess</Class>
<BaseURL>http://server_host:port/root_path</BaseURL>
<Init>
<BaseURL>http://server_host:port/root_path</BaseURL>
</Init>
</HttpAccessPlugin>
</MP4>
```

Custom implementations for the http access plugin are sandboxed at the Stream Container Level. You should copy them to the lib folder inside the configuration folder of the Stream Container.

Consider a scenario where a plugin that provides access to CQ CMS is created with the implementation class name com.my.httpaccess.CQAccess and packaged in the jar file CQPlugin.jar. If you use this plugin for your stream container default, the directory structure and configuration will look like this:

```
/conf
/http
 /origin.xml    <- Global configuration
 /containers
 /_default_
 /container.xml <- Per container configuration
 /lib
 /CQPlugin.jar  <- CQ Access Plugin
 /streams       <- Modules for this tenant
 /channel1
 /vod.xml       <- Per stream configuration
 ...
```

In this case, vod.xml will have the following sample configuration:

```
<Config>
 ....

<!-- Config for MP4 JIT Packaging -->
 <MP4>
```

```
 .....
 <HttpAccessPlugin>
   <Class>com.my.httpaccess.CQAccess</Class>

   <Init>
   <!-- Config for CQAccess plugin -->
   </Init>

 </HttpAccessPlugin>
</Config>
```

Cue Signal Format for HLS Manifest

Primetime Offline Package lets you add cue information in the JIT created m3u8. Adobe DPI format is supported for Ad Cues.

The format of cue information to be added in the m3u8 is determined using the option Config/CueMode, available for both Live and VOD origin modules. The valid values for this configuration option are the following:

PT_1_0	Used to generate cue information in PT 1.0 format as described in *Legacy HLS AD Cue Format* specifications for m3u8.
DPISimple	Used to generate cue information in DPI Simple mode as described in the *Primetime Digital Program Insertion Signaling specification*.
DPIScte35	Used to generate cue information in DPI SCTE mode as described in the *Primetime Digital Program Insertion Signaling specification*.

The DPI-SCTE mode is used if no cue mode is specified in the command line option.

Parsing DPI cues

Primetime Origin server supports parsing of cues generated by Primetime Live Packager. The supported cue formats are specified in the DPI specifications.

The Primetime Origin server processes SCTE 35 cues from the source HDS Manifest (F4M) and adds them to Just-in-time generated HLS manifest (M3U8). The format of cue information to be added in the m3u8 is determined using the option Config/CueMode, available for both live and VOD origin modules.

Web VTT support

Primetime supports repackaging of WebVTT files for HLS. WebVTT files are used to mark up external text track resources for your video files. You can create separate Web VTT streams (m3u8) from an input WebVTT file. The fragment duration for WebVTT segments is derived from the FragmentDuration setting under the MP4 element of the containing stream configuration.

The following is a sample variant playlist, assuming that ' english_en_full.vtt' file is available at 'vtt/origin' relative to the base location indicated in the MP4 element:

```
#EXTM3U
#EXT-X-MEDIA:TYPE=SUBTITLES,GROUP-ID="subs",
 NAME="English",DEFAULT=YES,
 URI="vtt/orig/english_en_full.vtt.m3u8",LANGUAGE="en"
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=150000,SUBTITLES="subs"
/vtt/video/movie.mp4.m3u8
```

The m3u8 for the VTT stream is shown below:

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:4
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-MEDIA-TIME:12.0
#EXTINF:4,english_en.12000.vtt
#EXTINF:4,english_en.16000.vtt
#EXTINF:4,english_en.20000.vtt
#EXTINF:4,english_en.24000.vtt
#EXTINF:4,english_en.28000.vtt
#EXTINF:4,english_en.240000.vtt
#EXT-X-ENDLIST
```

WebVTT does not support the following:

• Key Frame only streams
• Content protection/encryption
• Audio Only
• Ad cues

MP4 JIT Packaging

Origin Server supports the generation of just-in-time HDS or HLS VOD streams from an mp4 file that is copied to the local disk of Origin Server or located on the network.

For a VOD module on Origin Server, the following MP4 JIT transformations are available:

MP4 location	**JIT MP4 URI**
local disk	http://origin-server/container_name/module_name/mp4_relative_path/filename.mp4.f4m or http://origin-server/container_name/module_name/mp4_relative_path/filename.mp4.m3u8
on network	< HTTP-Base-URL >/ mp4_relative_path should be mentioned in the stream configuration at Origin. At the client side (player), the requesting URL is http: // origin - server / container _ name / module _ name / filename . mp4.f4m or http: // origin - server / container _ name / module _ name / filename . mp4.m3u8

An MP4 file is not directly accessible. If you access an MP4 file directly, a 403 error occurs. You can access MP4 file only through one of the following JIT transformations:

JIT Transformation available	**URI**
HLS	http://origin-server/container_name/module_name/mp4_relative_path/filename.mp4.m3u8
HLS Selected Tracks Only	http/origin-server/container_name/module_name/mp4_relative_path/filename.mp4.m3u8?tracks=id1,id2
HLS Key Frame Only (first video track)	http://origin-server/container_name/module_name/kfonly/mp4_relative_path/filename.mp4.m3u8
HLS Key Frame Only (specified video track)	http/origin-server/container_name/module_name/kfonly/mp4_relative_path/filename.mp4.m3u8?tracks=Video_Track_ID
HLS Key Frame Only from the Selected Tracks	http/origin-server/container_name/module_name/kfonly/mp4_relative_path/filename.mp4.m3u8?tracks=id1,id2

HLS Audio Only (first audio track)	http://origin-server/container_name/module_name/audio_only/mp4_relative_path/filename.mp4.m3u8
HLS Audio Only (specified audio track)	http://origin-server/container_name/module_name/audio_only/mp4_relative_path/filename.mp4.m3u8?trackid=Audio_Track_ID
HLS Audio only from the Selected Tracks	http://origin-server/container_name/module_name/audio_only/mp4_relative_path/filename.mp4.m3u8?trackid=id1,id2
HDS	http://origin-server/container_name/module_name/mp4_relative_path/filename.mp4.f4m
HDS Selected Tracks Only	http://origin-server/container_name/module_name/mp4_relative_path/filename.mp4.f4m?trackid=id1,id2
HDS Key Frame Only (first video track)	http://origin-server/container_name/module_name/keyframe_only/mp4_relative_path/filename.mp4.f4m
HDS Key Frame Only (specified video track)	http://origin-server/container_name/module_name/keyframe_only/mp4_relative_path/filename.mp4.f4m?trackid=Video_Track_ID
HDS Key Frame Only from the Selected Tracks	http://origin-server/container_name/module_name/keyframe_only/mp4_relative_path/filename.mp4.f4m?trackid=id1,id2
HDS Audio Only (first audio track)	http://origin-server/container_name/module_name/audio_only/mp4_relative_path/filename.mp4.f4m
HDS Audio Only (specified audio track)	http://origin-server/container_name/module_name/mp4_relative_path/filename.mp4.f4m?trackid=Audio_Track_ID
HDS Audio only from the Selected Tracks	http://origin-server/container_name/module_name/audio_only/mp4_relative_path/filename.mp4.f4m?trackid=id1,id2

The following configuration parameters have been added or modified in vod.xml file:

Name	Description
Config/MP4/FragmentDuration	Fragment duration in milliseconds to be used for MP4 JIT Packaging(both HDS and HLS). If not specified, default value of 4000 ms is used.
Config/MP4/TargetDuration	HLS Target duration in milliseconds to be used for MP4 JIT HLS Packaging. It's an optional parameter. In case it's not specified, target duration is set as the duration of the longest fragment. For details about target duration, refer HLS Spec at *http://tools.ietf.org/html/draft-pantos-http-live-streaming-10* . If the longest fragment created is longer than the target duration specified by user, then the longest fragment duration is used as the target duration.
Config/MP4/MaxIndexCacheSize	Approximate upper limit of the total size of the cached MP4-Index entries in bytes. This parameter should be set to a value less than the maximum heap size, for example half the heap size. The default value is 512MB.
Config/MP4/HTTP/HttpAccessPlugin/Class	Fully qualified class name of the Http Access Plugin to access remote mp4 files
Config/MP4/HTTP/HttpAccessPlugin/Init	XML Config for the Http Access Plugin must be specified under this element.
Config/Auth/SecurityToken/Key	The AES-128 bit key to be used for decrypting the PUT/POST token. This value shall be a string of 32 hexadecimal digits.
Config/Auth/SecurityToken/Timeout	Optional 'Timeout' for Token values in seconds. Default value is 600 seconds (10 minutes).
Config/Auth/DeleteSecurityToken/Key	The AES-128 bit key to be used for decrypting the DELETE token. This value shall be a string of 32 hexadecimal digits.
Config/Auth/DeleteSecurityToken/Timeout	Optional 'Timeout' for DELETE Token values in seconds. Default value is 600 seconds (10 minutes).
Config/CueMode	The format of cue info to be added in the JIT generated f4m/m3u8. The allowed values for this config option are:PT_1_0, DPISimple, DPIScte35. The default value is DPIScte35

HLS ingest

Origin Server 1.0 accepts HLS packaged by Primetime 1.0 packager. Origin Server 1.2 accepts HLS packaged by any source. It makes HLS available for clients through HTTP.

Supported input formats:

- .ts
- .acc
- .mp3

The m3u8 file published to the Origin Server should be in compliance with the HLS standards. Origin Server also accepts trick-play for HLS stream. In this case, segment entries in trick-play m3u8 are the byte-ranges of the segments of the main m3u8.

Disk management

For HLS ingest, the DiskManagementDuration configuration is ignored. Incoming m3u8 should specify the duration in which the fragments are retained on the disk. The related HLS specification mandates the following:

"When the server removes a media segment from the Playlist, the corresponding media URI SHOULD remain available to clients for a period of time equal to the duration of the segment plus the duration of the longest Playlist file distributed by the server containing that segment."

To meet this requirement, the duration in which fragments are retained in the disk is calculated using the following formula:

```
Retention duration of fragments = 2 * m3u8 sliding window duration + Target Duration
```

For trick-play m3u8 generated by encoder or external source, segment entries in the trick-play can be byte range references of the segments in the main m3u8. For such trick-play m3u8s, the following conditions must be met for disk management to work:

- The sliding window duration of both the trick-play m3u8 and the main m3u8 must be the same.
- The lead/lag time between the trick-play m3u8 and the main m3u8 must not exceed thrice the target duration.

HTTP Delete

The Origin Server also accepts the HTTP Delete requests to delete specified HLS segments. This is allowed only for HLS segments on a Stream module. HTTP Delete cannot be handled by VOD Module or for any other file type on Stream module.

If the Security Token is enabled on Origin Server, then HTTP DELETE requests need to carry valid authentication token. If the token is absent, the Unauthorized response code (401) is returned.

Byte Range Support

Origin Server accepts m3u8 where segment entries contain the #EXT-X-BYTERANGEtag. For such segment entries, the actual HTTP GET request for segments contains the Range headers. Origin Server doesn't honor such requests for syntactically valid Range headers. It sends a response with a status of 206 (Partial Content) containing the

applicable ranges of the entity-body. For such response, the Content-Range entity-header is sent and it specifies where in the full entity-body and the partial body should be applied.

If the byte-range-set is cannot be fulfilled, the server returns a response with a status of 416 (Requested range not satisfiable).

Origin Server supports only single byte range in the byte range set (including support for suffix byte range.) If more than one byte range is specified in the byte range set, Origin Server ignores the byte-range-set and serves the entire entity.

Cue Markers

Primetime Origin Server does not add or modify any cue marker information in the published m3u8 file. It is expected that the encoder or packager should add the relevant cue information in the m3u8.

Ingest of Encrypted HLS

Primetime Origin supports ingest of AES-128 encrypted HLS Stream. No JIT transformations will be performed on such streams. Origin Server only takes care of ensuring disk management of published segments as per the duration of published m3u8. Primetime Origin does not provide key hosting for AES-128 encrypted content. Key will have to be securely delivered through a separate key server which takes care of secure key delivery and client authentication.

Failover support

Origin Server does not modify the m3u8 posted to it. Encoder or packager publishing the m3u8 files need to ensure that m3u8 is HLS standard compliant. The encoder or packager should update the m3u8 file in a consistent manner.

Following are the constraints for consistent generation of m3u8:

- After a segment is assigned a sequence number, it cannot change.
- Sequence number should increase. Encoder or packager cannot introduce jumps in sequence number without removing all the previous entries from the m3u8 file.

Content-type header

For HTTP Push (PUT/POST) to Origin Server, it tries to determine the type of entity using either the file name pattern, extension, or the Content-Type header. If entity type cannot be determined for HTTP Push requests, then Origin Server returns the Unsupported Media Type (415) response code.

Following are the Supported Content-Type for determining entity type at the time of HTTP Push:

Content-Type Header	File Type
video/f4f	HDS Fragment
application/f4m+xml	HDS Manifest
application/vnd.apple.mpegurl	HLS Manifest
video/mp4	DASH Fragment
application/dash+xml	DASH Manifest

Following are the supported file extensions/name pattern for determining entity type at the time of HTTP Push:

File Extension/Name Pattern	File Type
.ts, .mp3, .aac	HLS Fragment
.bootstrap	HDS Bootstrap
.f4m	HDS Manifest
.m3u8	HLS Manifest
.drmmeta, .drm	DRM Metadata
.mpd	DASH Manifest
.m4s, .mp4	DASH Fragment
File Name Pattern <Prefix>Seg<Seg_Num>-Frag<Frag_Num>	HDS Fragment

Logging

Primetime Origin Server includes a log formatter that formats access logs in a manner that enables *Splunk* to easily index the logs. The formatter class is com.adobe.fms.http.KeyValueAccessFromatter.

To use this formatter for access logs, change the formatter property of com.adobe.fms.http.AccessFileHandler in logging.properties to com.adobe.fms.http.KeyValueAccessFromatter. As per *Splunk* logging best practices, this formatter logs all fields in key-value pairs with values enclosed in quote marks and different key-value pairs separated by commas.

The following are sample entries of the access log by this formatter:

```
date="2013-Oct-21 13:38:38.287 +0530",
resTime="11",
reqLen="0",
status="200",
resLen="191",
method="GET",
uri="/crossdomain.xml",
ip="10.40.62.12:33645"

date="2013-Oct-21 13:38:44.275 +0530",
resTime="103",
reqLen="0", status="200",
resLen="1691", method="GET",
uri="/_default_/_default_/livestream.f4m",
ip="10.40.63.232:55560"

date="2013-Oct-21 13:38:44.486 +0530",
resTime="155",
reqLen="0",
status="200",
resLen="15151",
method="GET",
uri="/_default_/_default_/livestream.drmmeta",
ip="10.40.63.232:55560"

date="2013-Oct-21 13:38:48.289 +0530",
resTime="2",
reqLen="0",
status="200",
```

```
resLen="191",
method="GET",
uri="/crossdomain.xml",
ip="10.40.62.12:33647"
```

The formatter uses the following key names:

Key Name	Description
Date	Log timestamp
resTime	Request processing time taken by Origin in milliseconds
reqLen	Request Length in bytes
status	HTTP response status code for the request
resLen	Response Length in bytes
method	HTTP Request Method
X-Forwarded-For	Client IP address extracted from the X-Forwarded-For header
uri	Requested URI
ip	Remote IP Address

The following are sample contents of the default logging.properties file provided with Adobe Primetime:

```
# Specify the handlers to create in the root logger (all loggers are children of the root
logger)
# The following creates two handlershandlers = java.util.logging.FileHandler,
java.util.logging.ConsoleHandler
# Set the default logging level for the root logger.level = INFO

#Uncomment this to enable FINE level logs for only com.adobe.fms and children loggers
#com.adobe.fms.level = FINE
#Settings for displaying log messages to
 Consolejava.util.logging.ConsoleHandler.formatter = java.util.logging.SimpleFormatter
 java.util.logging.ConsoleHandler.level = INFO
#Settings for writing log messages to origin.log
 java.util.logging.FileHandler.pattern = ./logs/origin.%g.log
 java.util.logging.FileHandler.level =  FINEST
 java.util.logging.FileHandler.formatter = com.adobe.fms.util.PTSimpleFormatter
 java.util.logging.FileHandler.append = true
# limit each file to 512MB before rolling over.
 java.util.logging.FileHandler.limit = 524288000
 java.util.logging.FileHandler.count = 4
#Settings for writing log messages to access.log
 com.adobe.fms.http.handlers = com.adobe.fms.http.AccessFileHandler
 com.adobe.fms.http.level = FINE
 com.adobe.fms.http.AccessFileHandler.pattern = ./logs/http/access.%g.log
 com.adobe.fms.http.AccessFileHandler.level = FINE
#use this KeyValueAccessFromatter to generate splunk-friendly access logs
#com.adobe.fms.http.AccessFileHandler.formatter = com.adobe.fms.http.KeyValueAccessFromatter
 com.adobe.fms.http.AccessFileHandler.formatter = com.adobe.fms.http.AccessFormatter
 com.adobe.fms.http.AccessFileHandler.append = true
 com.adobe.fms.http.AccessFileHandler.limit = 524288000
 com.adobe.fms.http.AccessFileHandler.count = 4
############################################################################################
## An example to debug log messages for the StreamContainer "foo"
## Setup a unique file handler
#com.adobe.fms.module.FileHandler1.pattern = ./logs/debugContainer.%g.log
#com.adobe.fms.module.FileHandler1.level = FINEST
#com.adobe.fms.module.FileHandler1.formatter = java.util.logging.SimpleFormatter
#com.adobe.fms.module.FileHandler1.append = true
#com.adobe.fms.module.FileHandler1.limit = 524288000
#com.adobe.fms.module.FileHandler1.count = 4
```

```
##com.adobe.fms.StreamContainer.foo.handlers = com.adobe.fms.module.FileHandler1
#com.adobe.fms.StreamContainer.foo.level = FINEST
##############################################################################################
## An example to debug log messages for the Stream "stream1" in the
## StreamContainer "foo" while using com.adobe.fms.module.FileHandler2
### Setup a unique file handler
#com.adobe.fms.module.FileHandler2.pattern = ./logs/debugStream.%g.log
#com.adobe.fms.module.FileHandler2.level = FINEST
#com.adobe.fms.module.FileHandler2.formatter = java.util.logging.SimpleFormatter
#com.adobe.fms.module.FileHandler2.append = true
#com.adobe.fms.module.FileHandler2.limit = 524288000
#com.adobe.fms.module.FileHandler2.count = 4
#com.adobe.fms.StreamContainer.foo.Stream.stream1.handlers = com.adobe.fms.module.FileHandler2
#com.adobe.fms.StreamContainer.foo.Stream.stream1.level = INFO
#
```

To use the Splunk formatter, uncomment the following line in the default logging.properties file:

```
#com.adobe.fms.http.AccessFileHandler.formatter = com.adobe.fms.http.KeyValueAccessFromatter
```

In addition, comment the following line in the same file:

```
com.adobe.fms.http.AccessFileHandler.formatter = com.adobe.fms.http.AccessFormatter
```

Error codes

GET requests from the clients

For HTTP GET requests from the client, the path on the disk is constructed from the base URL. Origin Server doesn't allow to access the folder while serving an HTTP GET request from the client.

The following are error messages returned by Origin Server.

Error message	Description
HTTP 404	Content doesn't exist.
HTTP 403	Forbidden access to the folder.
HTTP 500	Internal server error (any other server error)

PUT request from the packager

Packager uses HTTP PUT to publish the content to Origin Server. The file location in the disk is decided by the base URL. Origin Server serves only the files corresponding to successful PUT requests. Origin Server allows a PUT request over a successful PUT request happened before. In this case, if the PUT request is successful, the existing file is overwritten. If the PUT request is failed, it doesn't replace the original file.

Origin Server returns the following errors for PUT requests:

Error message	Description
HTTP 401	Unauthorized - request is without a security token or with an invalid or expired security token.
HTTP 403	Forbidden - the folder name already exists.
HTTP 500	Internal Server error - any other error. For example, the length of a PUT request exceeded the <*MaxContentLength*> that is configured.

Error message	Description
HTTP 507	Insufficient Storage - exceeded the storage limits and Origin Server is not able to clean up in time.