MSOD: Stream Packaging User Guide

March 03, 2022

Contents

Welcome to Media Services On Demand: Stream Packaging

Media Services On Demand (MSOD): Stream Packaging allows use of various on demand file formats, such as FLV and MP4, with RTMP ingress and egress in either Adobe HDS, Apple® HLS, or progressive FLV delivery (HD Flash 1.0) outputs.

ⓘ **Note:** HD Flash 1.0 is only supported for existing customers of this product.

MSOD: Stream Packaging uses dynamic streaming to provide a high-quality video experience for end-users. It also offers support for the broadest range of video end points to maximize content reach.

Dynamic streaming is an enabling technology that automatically selects and streams the highest quality bit rate to a video end point based on connection speed and device capabilities. If the end user's connection speed changes, the player responds by requesting a different bit rate from MSOD and a seamless bit rate switch occurs.

For example, if the available bandwidth on an end user's connection drops from 2 Mbps to 1 Mbps, the player detects the change and requests the stream of a lower bit rate to avoid rebuffering or video stutter. If the connection speed increases, and the available bandwidth returns to 2 Mbps, the player asks MSOD to resume streaming at the original 2 Mbps rate.

All of this complexity is hidden from the end user and the result is a consistent, high-quality viewing experience.

Encoding guidelines

For general encoding guidelines, refer to the On Demand Encoding Best Practices document, available on Control Center.

Supported codecs/containers

HDS/HD Flash 1.0 outputs

The following table lists the codecs and containers that are supported by Adobe HTTP Dynamic Streaming and HD Flash 1.0.

Container	Video codec	Audio codec	Comments
FLV	H.264 VP6 Sorenson Spark (H.263)	MP3[1] AAC PCM Nellymoser	Video-only works as well. For Nellymoser, only the 8 khz and 16 khz mono sound formats are supported

[1] MP3 audio format is supported only if wrapped in a MP4 container (applies to HD Flash 1.0 and Adobe HTTP Dynamic Streaming).

Container	Video codec	Audio codec	Comments
F4V	H.264	AAC	
MP4	H.264	AAC MP3	Audio-only (AAC) or Video-only also works
F4F/F4M	H.264 VP6	AAC MP3	

HLS output

The following table lists the codecs and containers that are supported by Apple HTTP Streaming.

Container	Video codec	Audio codec	Comments
F4V/MP4	H.264 H.264 Baseline Profile Level 3.0[2] Main Profile Level 3.1[3]	AAC-LDC up to 48 kHz, stereo audio	Audio-only (AAC) or Video-only also works

Following are links to web pages containing the video playback specifications for Apple iPad®, Apple iPhone®, and Apple iPod touch® devices, which list the types of encoding each device supports.

iPad	*http://www.apple.com/ipad/specs/*
iPhone	3G—*http://support.apple.com/kb/SP495* 3GS—*http://support.apple.com/kb/SP565* 4G—*http://www.apple.com/iphone/specs.html*
iPod Touch	2nd generation—*http://support.apple.com/kb/sp496* 3rd generation—*http://support.apple.com/kb/SP570* 4th generation—*http://support.apple.com/kb/SP594*

For more Apple HTTP Streaming encoding recommendations, refer to Apple's web page Best Practices for Creating and Deploying HD Live Streaming Media for the iPhone and iPad (*http://developer.apple.com/library/ios/#technotes/tn2224/_index.html*). Additionally, Apple has a web page with links to streaming documentation, including the latest best practice recommendations for encoding, deployment, app development, and app submission at *https://developer.apple.com/resources/http-streaming/*.

[2] iPhone and iPod Touch
[3] iPhone and iPad

Origin server requirements

If you host your videos on your own server, you must have adequate resources provisioned so that servers can retrieve the videos. These are the requirements for origin servers:

- Servers must be HTTP 1.1 compliant to support byte-range GET requests.
- Servers must return a Last-Modified header on every byte range GET request; the header must be identical for all chunks of a given file and must not change unless the file is replaced.

  ⓘ **Note:** If your origin consists of multiple servers behind a load balancer, or if the origin host name returns multiple server IP addresses, you must ensure that, for a given file, the Last-Modified header is identical on each server and IP address.

- Servers must return an updated **Last-Modified** header when a file is replaced. You must also purge the file from the Edge servers to ensure that they fetch the new version.
- Servers must have adequate processing power and provisioned bandwidth to handle the maximum expected simultaneous requests from the servers

  ⓘ **Note:** This will never exceed, and is usually substantially less than, the maximum number of requests received when not using Media Services On Demand.

You need not maintain your own origin server if you use Akamai NetStorage.

Open issues

The Speex audio compression format is not supported.

Truncated FLV files are not currently handled.

I-frame playlists are not supported in dynamic stream packaging.

Create configurations

Before you can create new streams, you must create at least one stream packaging configuration to establish parameters for packaged streams.

You can create as many configurations as you have CP codes available, depending on how you would like to have your streams being reported on and billed.

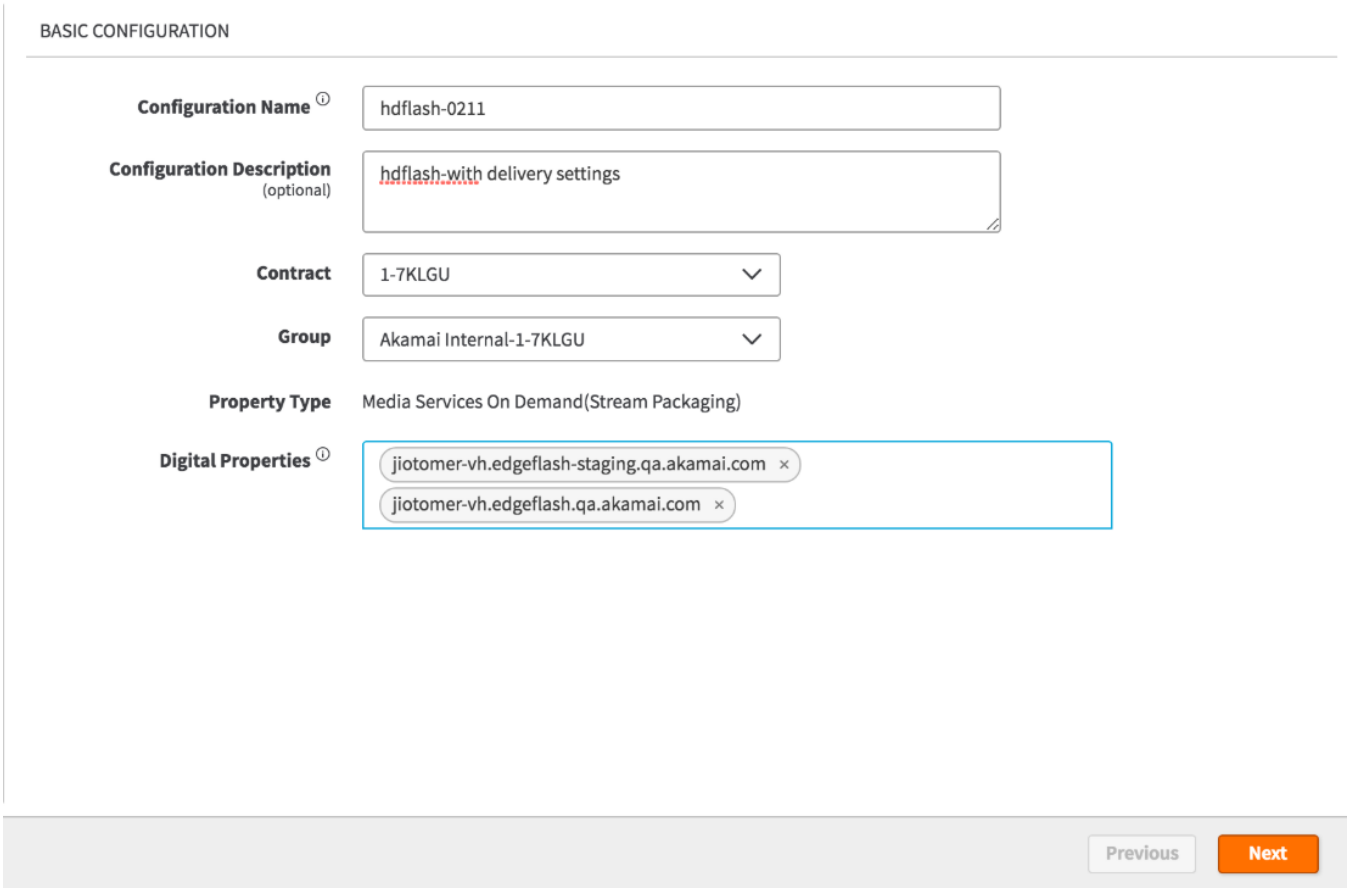
1. Access https://control.akamai.com/.
2. Log in using an account that's been provisioned for access to MSOD Stream Packaging.
3. Open the application. Go to > **MEDIA** > **Other Media services** > **On-Demand Stream Packaging configurations**.

What's next?

Begin the creation process and *Specify basic settings* on page 5.

Specify basic settings

Click **Add Configuration** on the MSOD stream packaging configurations page to display the basic settings page.



1. In the Configuration Name text box, enter a name for the configuration file.The configuration file is stored as an XML file, and the name you choose should typically reflect the digital properties it includes (for example, streampackaging.example.com.xml).
2. Optionally enter a description for your configuration.
3. In the **Digital Properties** text box, enter the domain name or hostname for the configuration. For example, **streampackaging.example.com**.

   You must include a hostname in the form of ***-vh.akamaihd.net**. For example, **customername-vh.akamaihd.net**. This will be created behind the scenes when you complete your configuration creation. Make sure that there is one and only one hostname ending in **-vh.akamaihd.net**.

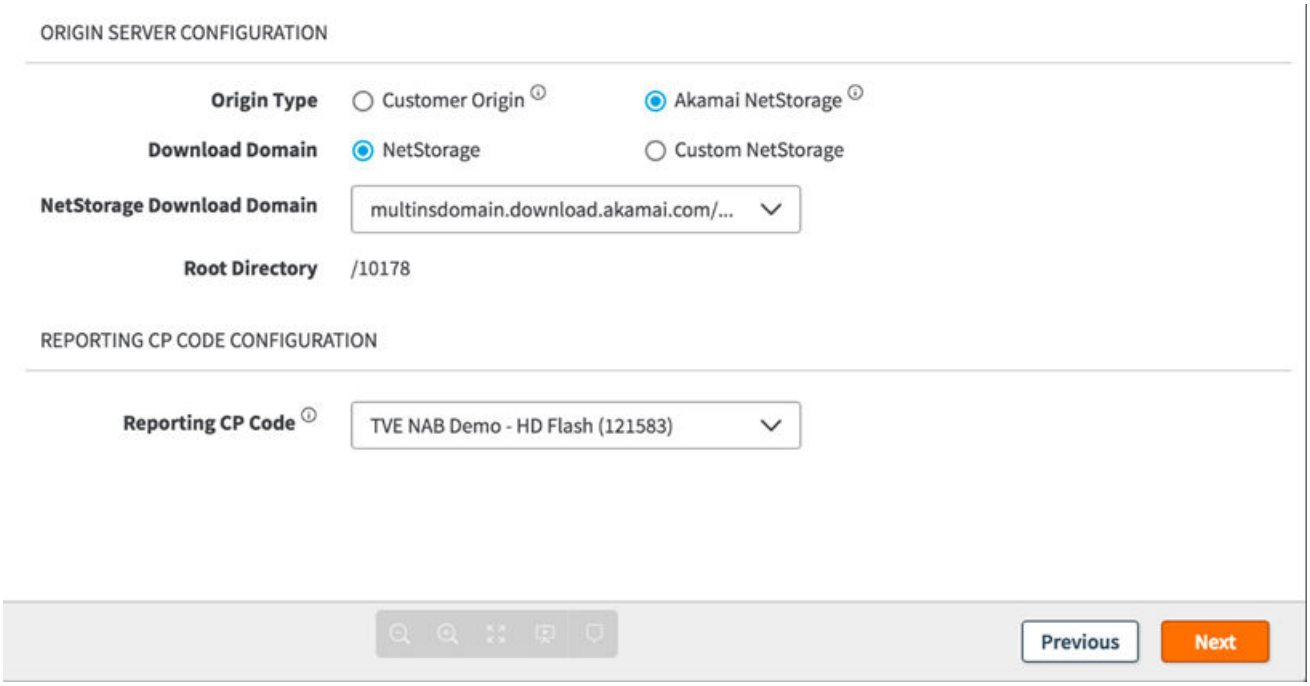
   Staging digital properties for your **akamaihd.net** hostname will be automatically added and can be tested by activating configuration on the Staging network. For example, if your hostname is **customer-vh.akamiahd.net** then the corresponding staging hostname would be **customer-vh.akamiahd-staging.net**.

   Note: The **-vh** suffix indicates it is associated with Media Services On Demand Stream Packaging, for example, **example-vh.akamaihd.net**. This suffix supersedes the previously-used **-f** and **-i** suffixes. However, if you have older configurations that use either of these, they will continue to operate normally.

If you have more than one digital property, you can add additional digital properties clicking **+Add Another Additional Digital Property**. The limit is 10. If you need more digital properties, contact your Account Representative.

Specify origin and CP code

Click **Next** on the basic settings page to display the origin and CP code page.



Under origin server configuration, specify the following:

1. Select **Customer Origin** or **Akamai NetStorage** as the origin type, depending on what you plan to use as an origin for your on demand content.
2. Select **NetStorage** as the download domain for Akamai NetStorage origin type. Select **Custom NetStorage** for Customer Origin. See *Origin server requirements* on page 4 for information on hosting your own content.

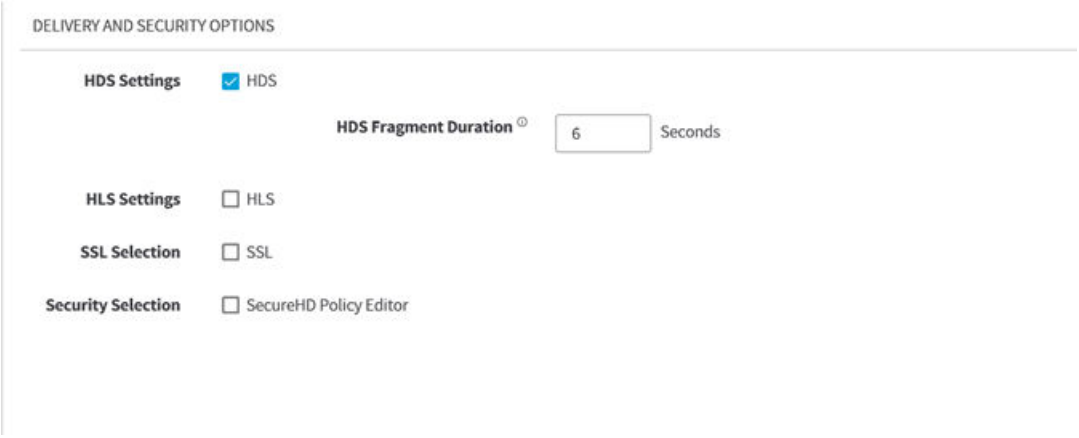
Specify delivery and security options

Click **Next** on the origin and CP code page to display the delivery and security options page.

You can configure HDS settings, HLS settings, SSL selection, and security selection on this page.

Configure HDS settings

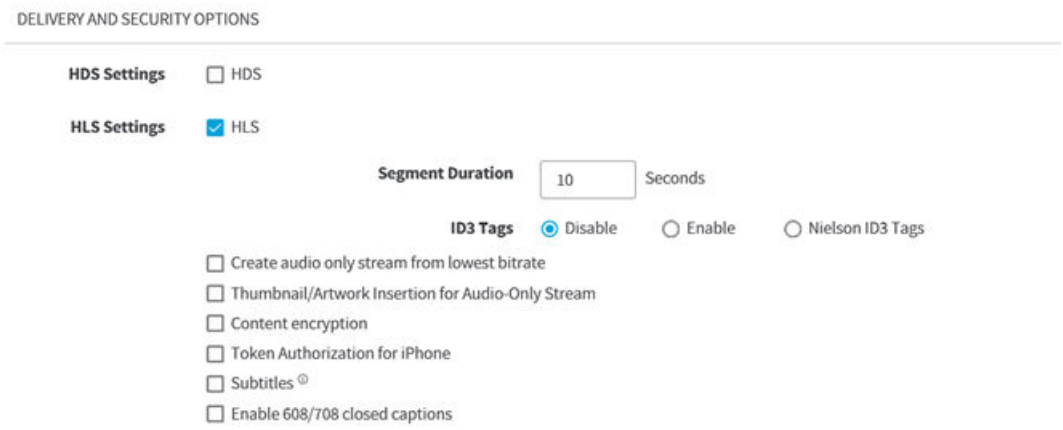
Select HDS settings in the delivery and security options page.



Specify a value (in seconds) for the HDS Fragment Duration. This value should be equal to the keyframe interval or multiples of it. The default value is six (6) seconds, which would work for one-, two-, three-, and six-second keyframe intervals. If your interval is different from these values, set the appropriate value in this field. Setting this value incorrectly could cause undesirable bit rate switch downs in HDCore versions lower than 2.8.

Configure HLS settings

Select HLS Settings in the delivery and security options page to display the HLS settings options.



Specify segment duration

In the Segment Duration field, enter a duration (between 2 and 60 seconds) for your segments or fragments. The default duration is 10 seconds. If you change the HLS segment duration for an existing VOD stream packaging configuration, ensure that you purge any content from the cache.

Specify ID3 tags

ID3 tags are disabled by default.

Select one of the following:

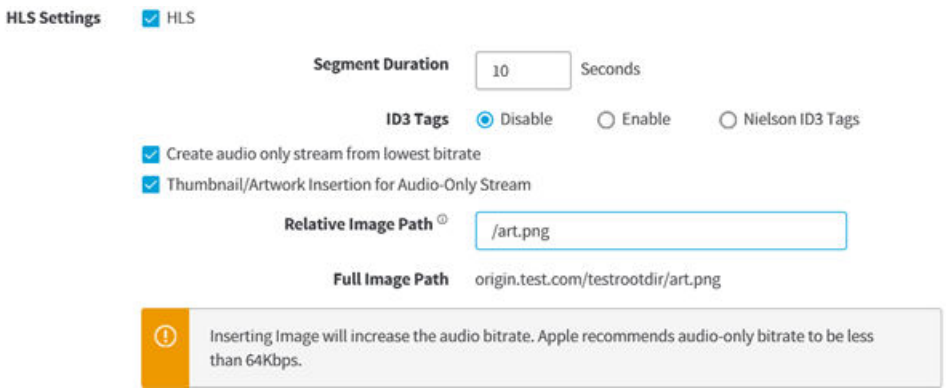
- Click **Disable** if you don't want to specify any ID3 tags.
- Click **Enable** to optionally activate timed metadata using options in the ID3 Tags area.Timed metadata is custom metadata that is added to the source content for passing the following: Cue points, timecodes, ad markers, an album name, a title — ID3 tags.
- Click **Nielsen ID3 Tags** to enable program ratings using Nielsen ID3 tags. For details about Nielsen ID3 tags (rating information), see *http://www.nielsen.com*.

Create audio only stream from lowest bitrate

If you want automatically extract an audio-only stream from your lowest published bit rate, select **Create an audio only stream from lowest bitrate**. If you are publishing an audio-only stream, and you select this option, your stream takes precedence over that produced by Media Services On Demand.

Thumbnail/Artwork insertion for audio only stream

Select **Thumbnail/Artwork insertion for audio only stream** to to display an image when the audio-only stream is played.



Enter the path of the image that you would like to display in the Relative Image Path field. MSOD supports only jpg and png format files of 40 KB or less.

Configure content encryption

Select Content encryption to deliver encrypted content from the edge servers to the player runtime. Encryption uses AES-128 bit crypto algorithm per Apple's specification.

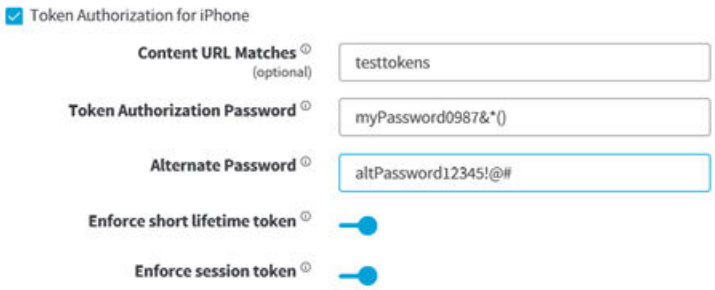
Note: You cannot use content encryption along with the secure HD policy editor.

The encryption key in 32 hex digits is automatically filled in the Auto generate key field.

Enter the encryption key URL to provide to clients in the playlist file. For example: `http://foo.bar.baz/x/y/z`

Configure token authorization for iPhone

Select **Token Authorization for iPhone** to enable token-based authentication for iPhones. This authenticates the users who attempt to log in to a server, a network, or some other secure system, using a security token provided by the server.

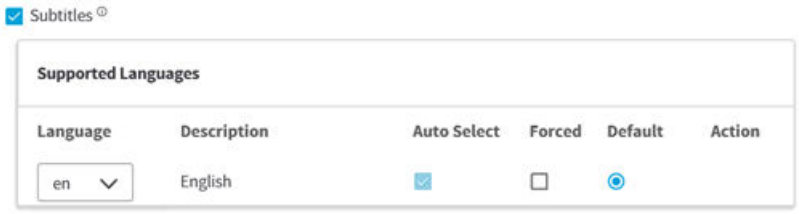


Enter the following:

1. Enter a string to match the requested URL against when using token authorization. Use * to specify wildcard matches. For example, if you enter `/protected/*`, only content in this folder is protected by token authorization. If left blank, all content will have token authorization enforced.
2. Enter a long and complex string as the token authorization password.
3. Enter a long and complex string as the alternate password, which is used temporarily when you change the token authorization password.
4. Use the slider to (optionally) select enforce short lifetime token on initial player requests.
5. Use the slider to (optionally) select enforce session token for all ongoing requests, including playlists, content segments, and keys.

Configure subtitles

Select **Subtitles** to configure subtitles in your HLS stream.



There should be either DFXP or WebVTT file per language, hosted in the same storage location as the media asset, if in a different location should be specified with the appropriate query string.

File should be named as per the naming convention consisting of language code and media identifier, along with underscore '_' separator. Default media identifier will be the same as the media asset name, Media identifier can be overridden in a query string. If the file name follows different naming convention then the complete file name will be specified in a query string.

1. Select a language for the subtitles. The description automatically appears.
2. Select Auto Select or Forced.

Use a DFXP/WebVTT file per each language that needs to be located in the same storage as your media asset. If it is stored in a different location, you must use query strings to set your subtitles as described in the section *Examples of setting up Subtitles via UI* on page 24.

If you are setting up your subtitles on UI, use the following functions:

- Select the input file format (the **VTT** or **DFXP** radio button) and file naming convention (the **Prefix Identifier** or **Suffix Identifier** radio button). The names of the DFXP/WebVTT file should consist of a media asset and a corresponding language code separated by an underscore.

  You can change the file naming convention in some of your output during playback using the query strings as described in the section *Query strings for setting up subtitles* on page 26.

- Enable the default for subtitles' language and select this default using the scroll-down menu and the **DEFAULT** radio button. Only one default language can be allowed. The subtitles in the default language start on playback. After you select your default language, it will automatically be added to the list of the autoselected languages.

- Add another language for your subtitles using the **Add Another** option and the **LANGUAGE CODE** scroll-down menu.
- Add a customer language (which is not on the preset options of the **LANGUAGE CODE** scroll-down menu) for your subtitles using the **Add Customer Language** option.

  Note: You can have up to 15 languages on the list. As soon as you add the fourth language, the **Add Another** and **Add Customer Language** options disappear from UI.

- If the **AUTOSELECT** option is turned on for a certain language, and the **Auto (Recommended)** option is selected during playback, this language appears as an available subtitle choice during playback.
- If the **FORCED** option is selected, this language is automatically turned on for your video asset.

  Note: Some players do not honer the **FORCED** option.

For examples that illustrate how the enabled **DEFAULT**, **AUTOSELECT**, and **FORCED** options affect your playback, refer to *Examples of setting up Subtitles via UI* on page 24.

Use closed captioning

To enable closed captioning, select **Enable 608/709 closed captions**.

Closed captions is a process that allows insertion of data or text that corresponds to the video scene. This can be done to provide accessibility to hearing-impaired persons (close captions or open captions), language translations for multi-region programming, or any other reason a producer chooses.

You can pass closed captions to a viewer in one of these ways:

- Embedded into the video file itself.
- Delivered in an accompanying *sidecar* file.

Note: In the h.264 input stream, if the subtitles text is present in the NALU header type 6 (SEI) as ATSC picture user data, it will be passed through and available in the HLS output.

Select SSL

Before you configure SSL, ensure that the origin supports SSL. Select **SSL** to configure end-to-end SSL from edge to origin. This prevents protocol downgrades throughout the path and mixed content warnings.

Activate your configuration

The Edge Staging Network (ESN) provides an environment to test your configurations without impact on your live, production traffic. After successful testing on ESN, activate the same configuration file version for production. Most standard current-version production features can be tested on ESN, but note that ESN is for testing functionality and not load, stress, or performance.

Contact your Akamai representative to review your configuration for any of the following incompatible features:

Global Traffic Manager GTM (*.akadns.net) Two Tree configurations SiteShield Additional information and training is available in the ESN User Guide and the On Demand Training Module.

Akamai Staging hostnames (**customer-vh.akamaihd-staging.net**) corresponding to your hostname (**customer-vh.akamaihd.net**) will be added for all staging activations for Universal Streaming. You can test these changes on Akamai Staging networkk before activating the configuration on production.

Set up the video playback for HDS and HD Flash 1.0 outputs

Media Services On Demand playback applications for HDS and HD Flash 1.0 outputs are built using provided Adobe ActionScript® 3.0 code library called **HDCore**. The purpose of this library is to make it easy for developers to add Media Services On Demand functionality to existing players.

You can build players for Media Services On Demand: Stream Packaging using any IDE that allows you to author in ActionScript 3.0 and compile to Flash Player 10. The core library is pure ActionScript 3.0 and is not dependent on the Adobe Flex® framework or any other framework for implementation. The most common development environments are Adobe Flash Builder® and Adobe Flash Designer.

Media Services On Demand delivers HD Flash 1.0 FLV files progressively to a Flash Player, version 10.0 or greater. Progressively here means over a network connection and through a null **NetConnection**. Contrary to the restrictions most associated with progressive delivery, you can instantly seek any point in these files, smoothly switch between different renditions of the content at alternate bit rates, and play On Demand content. These FLV files can actually wrap H.264 content, and in fact, the majority of content delivered over the network is H.264.

Helpful definitions for HDS output

This section introduces you to some key terms and definitions.

File type	Description	Extension	Example
Flash Media Manifest File	This file is requested by the player once at the start of the play session. It contains a list of all the available bit rates (tracks) for the requested stream along with the DVR window and other related metadata. The player uses the bit rates in this file to make switching decisions based on available bandwidth.	.f4m	manifest.f4m
Fragment File	These are small files suited for Media Services On Demand that contain the audio and/or video media data in the MP4 format.	N/A	1200_2d7ba93e14ed84fc-p_Seg1-Frag5456490

Use Media Services On Demand player components

As previously stated, Media Services On Demand playback applications for HDS and HD Flash 1.0 outputs are built using a Media Services On Demand provided ActionScript 3.0 code library called **HDCore**. The purpose of this library is to make it easy for developers to add Media Services On Demand functionality to existing players.

- **HDCore**

  - There are two core classes within this library:

    com.akamai.hd.ZStream — is used for HDS output.

    com.akamai.hd.HDNetStream — is used for HD Flash 1.0 output.

  - Full ASDOCS are available for each of these classes and are included with any code distribution. The code itself is distributed as a SWC library.

  - HDCore 2.3+ integrated into a framework built on Flash Player 10.1 and above is the minimum version required for HDS output.

  - Media Analytics integrated into HDCore:

    HDCore, version 1.1+ is the minimum version required. If you wish to use Media Analytics without integrating it into HDCore, you can use any HDCore version you like.

- **OSMF Plugin**

  - Advanced Streaming Plugin for OSMF is not required in order to use Adobe HTTP Dynamic Streaming. If you choose, however, to use the OSMF player and also leverage all Media Services On Demand features, this plugin provides support for it; version 1.0.4+ is the minimum version that supports it. For more details, refer to *Advanced Streaming Plugin for OSMF* available on Control Center.

    ⓘ **Note:** Usage of Player Components is enforced in all cases in which Adobe HTTP Dynamic Streaming is used.

Use other compatible playback methods

- LongTail Ad Solutions' JW Player® component

  For details, refer to *JW Player Plugin User's Guide*.

- Flowplayer, Ltd.'s flow player® video player component

  For details, refer to *Advanced Flowplayer Provider User's Guide*.

Use Media Services On Demand support players

Media Services On Demand: Stream Packaging provides support video players that you can use to test and troubleshoot your streams. Access them at the following locations:

- HDS output:

  HDS Test Player

- HD Flash 1.0 output:

  Flash HD Test Player

Serve your content

SMIL files and Client-Side URL syntax provide definitions to end-users' players regarding the multiple bit rates that are available for a given On Demand asset. Following are the situations in which you can use each, based on container type (these can be served from either NetStorage or your own Origin):

FLV	SMIL = YES Client-Side URL Syntax = YES
F4V/MP4	SMIL = YES Client-Side URL Syntax = YES
F4F	SMIL = NO Client-Side URL Syntax = NO *F4M = YES

*F4M is generated during the packaging process.

Using SMIL files

When streaming multiple bit rate assets, your Flash playback applications make use of SMIL files to map the available bit rate to the appropriate On Demand files. Following are some examples of SMIL files used with On Demand content.

Option 1: "vod" player feature

Use the "vod" feature in the player SMIL. This is the easiest and most foolproof way to do things and is almost always the right option.

example1a.smil:

```
<?xml version="1.0"?>
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN" "http://www.w3.org/2001/
SMIL20/SMIL20.dtd">
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
<head>
<meta name="title" content="This Is My Stream" />
<meta name="HDBase" content="http://example.akamaihd.net/" />
<meta name="vod" content="true" />
</head>
<body>
<switch id="whatever">
<video src="video.300k.mp4" system-bitrate="300000"/>
<video src="video.500k.mp4" system-bitrate="500000"/>
<video src="video.800k.mp4" system-bitrate="800000"/>
<video src="video.1000k.mp4" system-bitrate="1000000"/>
</switch>
</body>
</smil>
```

Note that **HDBase** must include the protocol (**HD** or **HDs**) and the domain name, but no additional path components. Also, all subdirectories must be listed at the beginning of each URL (See example 1b).

> **Note:** The configuration typically prepends certain path elements when requesting content from NetStorage such as the top-level CP code directory (for example, /9389). It is unnecessary, therefore, to explicitly add these path components in the URL used in the SMIL file.

All SMIL files referencing the same streams should make all of the bit rates available for maximum caching efficiency, since each stream is actually cached separately at the Edge if it is referenced from a different complete set. The HDCore classes provide a mechanism to limit the maximum bit rate that the player chooses if you do not want a given player session to have the option of switching up to the highest bit rate(s).

This example illustrates how to include a subdirectory in an On Demand SMIL file.

example1b.transformed.smil:

```
<?xml version="1.0"?>
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN" "http://www.w3.org/2001/
SMIL20/SMIL20.dtd">
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
<head>
<meta name="title" content="This Is My Stream" />
<meta name="HDBase" content="http://example.akamaihd.net/" />
<meta name="vod" content="true" />
</head>
<body>
<switch id="whatever">
<video src="subdir/video.300k.mp4" system-bitrate="300000"/>
<video src="subdir/video.500k.mp4" system-bitrate="500000"/>
<video src="subdir/video.800k.mp4" system-bitrate="800000"/>
<video src="subdir/video.1000k.mp4" system-bitrate="1000000"/>
</switch>
</body>
</smil>
```

> **Note:** The content paths here do not need to be identical except for the bit rate specification. However, the player's behavior is to construct a single URL containing all paths in the SMIL file, and the shorter the length of the common prefix and suffix across all of the paths, the shorter will be the resulting URL. An excessively long internal URL may exceed limits within the player, Flash runtime, browser, or Media Services On Demand, causing difficulty with playback.

Option 2: Server-side SMIL

You can also use a "server-side SMIL" to associate a set of bit rates with a short ".ssmil" URL, which is then referenced from a client-side SMIL.

> **Note:** This option is much more error prone and complex, so option 1 is almost always preferred.

This example consists of `example2a.ssmil`, which is uploaded onto the origin as `/subdir/example2a.ssmil`, and `example2a.smil`, which is made available separately; the latter's URL is the one sent to the player.

example2a.ssmil

```
<?xml version="1.0"?>
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN" "http://www.w3.org/2001/
SMIL20/SMIL20.dtd">
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
<head>
<meta name="title" content="This Is My Stream" />
<meta name="HDBase" content="/subdir/" />
</head>
<body>
<switch id="whatever">
<video src="video.300k.mp4" system-bitrate="300000"/>
<video src="video.500k.mp4" system-bitrate="500000"/>
<video src="video.800k.mp4" system-bitrate="800000"/>
<video src="video.1000k.mp4" system-bitrate="1000000"/>
</switch>
</body>
</smil>
```

example2a.smil

```
<?xml version="1.0"?>
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN" "http://www.w3.org/2001/
SMIL20/SMIL20.dtd">
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
<head>
<meta name="title" content="This Is My Stream" />
<meta name="HDBase" content="http://example.akamaihd.net/" />
</head>
<body>
<switch id="whatever">
<video src="subdir/example2a.ssmil/bitrate=300000" system-bitrate="300000"/>
<video src="subdir/example2a.ssmil/bitrate=500000" system-bitrate="500000"/>
<video src="subdir/example2a.ssmil/bitrate=800000" system-bitrate="800000"/>
<video src="subdir/example2a.ssmil/bitrate=1000000" system-bitrate="1000000"/>
</switch>
</body>
</smil>
```

The ".ssmil" file sets `HDBase` as the object's path only. You can also set `HDBasemerri` to `/` and then repeat `/subdir/` in each video `src` tag in the .ssmil. The client-side ".smil" file still must include the `subdir` element as part of each individual video's `src` tag.

A server-side SMIL must have the extension ".smil" or ".ssmil" for the On Demand system to treat it as a SMIL file instead of a single-bit-rate video file.

The files in Example 2a can also be combined by renaming the `HDBase` and `src` in example2a.ssmil to `server-HDBase` and `server-src`, respectively, and then combining the `video` tags together. Note that the default configuration will not serve the ".smil" file to the end user from "example.akamaihd.net," so either the same file must be hosted in the On Demand origin location, as well as in another location, or a custom configuration must be crafted to serve the ".smil" files (but not other files) from the common origin location

under a different hostname or something similar. Your Account Representative can assist in crafting this configuration if it is required.

example2b.smil:

```
<?xml version="1.0"?>
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN" "http://www.w3.org/2001/
SMIL20/SMIL20.dtd">
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
<head>
<meta name="title" content="This Is My Stream" />
<meta name="HDBase" content="http://example.akamaihd.net/" />
<meta name="server-HDBase" content="/subdir/" />
</head>
<body>
<switch id="whatever">
<video src="subdir/example2b.smil/bitrate=300000" server-src="video.300k.mp4"
system-bitrate="300000"/>
<video src="subdir/example2b.smil/bitrate=500000" server-src="video.500k.mp4"
system-bitrate="500000"/>
<video src="subdir/example2b.smil/bitrate=800000" server-src="video.800k.mp4"
system-bitrate="800000"/>
<video src="subdir/example2b.smil/bitrate=1000000" server-
src="video.1000k.mp4" system-bitrate="1000000"/>
</switch>
</body>
</smil>
```

Using client-side URL syntax

If you are unable to generate or provide server-side SMIL files, you can alternatively use a client-side URL format with which you can encode the entire SMIL contents in a single URL. For Media Services On Demand (Stream Packaging), the client-side URL takes the form:

```
http://<config_name>-vh.akamaihd.net/z/<subdirectory>/
<common_filename_prefix>,
<bitrate>,<bitrate>,<bitrate>,<bitrate>,common_filename_suffix>.csmil/
manifest.f4m
```

So a published URL to the device for this form of integration might look like:

```
http://example-vh.akamaihd.net/z/movies/example2a_,
300000,500000,800000,1000000,_event1.mp4.csmil/manifest.f4m
```

Using the above examples and assuming a CP code of 9389, your content would reside on NetStorage at example.download.akamai.com/9389/movies/ as:

- example2a_300000_event1.mp4
- example2a_500000_event1.mp4
- example2a_800000_event1.mp4
- example2a_1000000_event1.mp4

Note: If you plan to stream just a single bit rate file, it does not need client-side URL and could be added by itself. For example, a single mp4 file called hello.mp4 can have a syntax of <hostname>/z/hello.mp4/manifest.f4m.

Use query strings (for HDS output only)

There is one query string available for use with your client-side URL for HDS output.

Note: The values below are in kbps and are applied to the combined audio + video bit rate.

Bit rate filtering (b)

If you would like to filter out certain bit rates — to test a particular bit rate version, for example — you can do so by adding a `b` query string to the end of the publish URL. There are four uses:

- Filter all bit rates except those specifically designated (and the lowest audio rate); the string takes the form:

  ```
  b=bitrate,bitrate,bitrate,...
  ```

  So, for example, if you had bit rates 200, 300, 700, and 1200, and you wanted to include only bit rates 200, 700, and the lowest audio rate, your URL would appear as:

  ```
  uri/manifest.f4m?b=200,700
  ```

  This would produce only the 200 and 700 bit rate streams, as well as the lowest audio bit rate.

- Designate a bit rate range (in kbps) outside of which the stream or streams you wish to filter fall; the string takes the form:

  ```
  b=lower_end-upper_end
  ```

  So, for example, if you had bit rates 200, 300, 700, and 1200, and you wanted to include all bit rates at or greater than 200 and at or less than 700, your URL would appear as:

  ```
  uri/manifest.f4m?b=200-700
  ```

  This would produce the 200, 300, and 700 bit rate streams.

- Designate the low end of a bit rate range (in kbps) where all bit rates below the value are filtered; the string takes the form:

  ```
  b=low_end-
  ```

  So, for example, if you had bit rates 200, 300, 700, and 1200, and you wanted to include all bit rates at and higher than 300 kbps, your URL would appear as:

  ```
  uri/manifest.f4m?b=300-
  ```

This would produce the 300, 700, and 1200 bit rate streams.

- Designate the high end of a bit rate range (in kbps) where all bit rates at and above the designation are filtered; the string takes the form:

  ```
  b=0-high_end
  ```

  So, for example, if you had bit rates 200, 300, 700, and 1200, and you wanted to include all bit rates lower than or equal to 700 kbps, your URL would appear as:

  ```
  uri/manifest.f4m?b=0-700
  ```

  This would produce the 200, 300, and 700 bit rate streams.

- Any combination of the above.

  So, for example, if you had bit rates 200, 300, 700, 1200, and 1800, and you wanted to include all bit rates except 700, your URL could appear as:

  ```
  uri/manifest.f4m?b=0-300,1200-
  ```

  This would produce only the 200, 300, 1200, and 1800 bit rate streams, as well as the lowest audio bit rate.

Flexible playback (start | end)

This query string enables end-users to begin playback at a specific time (in seconds) within the VoD stream. For example:

```
uri/manifest.f4m?start=30&end=500
```

Note: The start time is mandatory but the end time is not. If the end time is not provided, the stream will play until the end of the VoD asset.

Fragment duration (sd)

This query string can be used to override a stream's fragment duration in case its keyframe interval and, hence, its fragment duration is different than what was set in the configuration (values should be designated as seconds (s) or milliseconds (ms). For example:

```
uri/manifest.f4m?sd=6006ms
```

```
uri/manifest.f4m?sd=6s
```

Test your content

You should test your content before making it available to your end-users to ensure a problem-free experience.

Note: Staging hostnames are added automatically for all **akamaihd.net** domains in order to test configurations on the Staging network. Staging hostnames will look as follows: **akamaihd-staging.net**.

Applying security

Security for HDS output

Security for HDS output is provided by the SecureHD Policy Editor (SPE), which is configurable through Control Center. For information regarding SPE and its configuration, refer to the *SecureHD Policy Editor User's Guide Network* available on Control Center.

Note: It is recommended that you do not have more than 10 hostnames associated with SecureHD Policy Editor for performance reasons. If you require more than 10 hostnames associated with SecureHD Policy Editor, contact your Account Representative for assistance.

Security for HD Flash 1.0 output

The following are security offerings for HD Flash 1.0 streams delivered with Media Services On Demand: Stream Packaging. Each item is disabled in the default configuration, so they must be explicitly enabled by your Account Representative.

- *Using Player Verification* on page 22 — Verifies that the SWF attempting to play the content is valid and authorized. This feature is similar in functionality to SWF Verification offered by Adobe's Flash Media Server; it is, however, an entirely different approach to SWF authentication and does not use Flash Media Server, or require an RTMP connection to be activated. If the SWF fails to pass the authentication test, the Edge server stops delivering content to it after five (5) seconds.
- *Using Token Authorization* on page 23 — Protects the initial content request. A customer-built token service generates a unique, single-use, public, time-delimited token for the content, based on a secret shared with Media Services On Demand. This token service only generates the token if the user is a valid user for the application. Examples might include LDAP or a cookie for successfully logging in to a site. If the token is incorrect or missing, no content is returned at all from the Edge servers.
- **GEO Blocking** — Allows you to control which world regions are able to view your stream.

Each security type can be implemented independently of the others or together in any combination. The following examines how to implement each of these.

Using Player Verification

You must ensure that only SWFs you have authorized are allowed to connect to your streams.

How to

1. Inform your Account Representative that you wish to activate Player Verification on your Media Services On Demand account so that the server side is correctly provisioned. They will also provide you details for accessing the auth bin on NetStorage. If you are already using SWF Verification for your RTMP delivery over Flash Media Server, this is the same bin.

2. Upload the uncompressed version of the Flash SWF (meaning the binary SWF is uncompressed) that you wish to authorize for Media Services On Demand. When you export a SWF from Flex Builder™ or Flash Builder, it is compressed by default. In Flash CS4 and CS5, it is also compressed by default, although there is an option to save it as uncompressed. Either way, to make the process simpler for you, a simple uncompression application is available for you — *SWF Authentication Utility*.
3. This online application allows you to upload your standard compressed SWF. It uncompresses it and then allows you to download it to your desktop. You can then upload it to the auth bin described in step 1 above.
4. In the case of players that dynamically load sub-SWFs at run-time, the SWF you should use for auth is the one that controls (or establishes) the STAGE object in your player. So if **a.swf** loads **b.swf**, which uses HDNetStream, the SWF you should upload for Player Verification is **a.swf**—the one at the top of the display stack.
5. The last action you need to take to enable Player verification is to set the **displayObject** property on your HDNetStream instance by passing it a reference to any object that participates in the display stack and that has access to the **stage** property. Exactly which object you pass is unimportant. Since HDNetStream extends NetStream, it itself has no visibility into the player, and so you must provide it via this property. If you fail to set this property prior to calling **play()**, Player Verification will fail, even if you have uploaded the correct uncompressed SWF to your auth bin.

What you should see

Once Player Verification is enabled, the Edge server will start delivering content and then will query the Flash client to authenticate itself. This process will take about 1.5 seconds. If the Edge server cannot successfully authenticate the SWF within 5 seconds, it terminates the connection. If authentication is successful, playback will continue uninterrupted. Verification repeats itself with each new **play()** request the HDNetStream class makes.

Using Token Authorization

To enable Token Authorization protection for your player and content, take the following steps.

How to

1. Inform your Account Representative that you wish to activate Token Authorization protection for your HD Flash 1.0 content. This ensures the server side is correctly configured.
2. Your Account Representative will provide you sample server-side scripts that you run to generate a token service. These scripts are available in all the common server-side languages (ASP, Perl, PHP, and Sun Microsystems® Java™ languages). There will be a shared secret (and an optional salt) that both Media Services On Demand and your service code share.
3. The service should apply whatever business logic your application needs to validate a specific user. For example, it could require a session cookie to be present in the request, which would indicate that the user has authenticated themselves against a portal or login gateway. It could also carry an explicit token that you hand to the player upon instantiation. You could filter by IP address if you wish to authorize use solely within a company LAN, etc.
4. When the Web service is built, you must create an ActionScript class that calls it. This class must implement the ITokenService interface. Simply put, this interface requires the class to offer up a **requestTokenizedURL()** method, which the HDNetStream class uses to request a tokenized URL and then two callback functions to call on both success and error. The callback functions may seem a bit antiquated, but the inability of ActionScript 3.0 to support events in interfaces necessitates their use.

5. In your player, you must then set the **requiresEdgeAuth** property to **true** before calling **play()**.
6. In the player, you must also instantiate an instance of this TokenService class and register it with the HDNetStream prior to the calling **play()**. This registration is done by setting the tokenService property to the token service instance you just created.
7. The first time you call **play()** for an asset, the HDNetStream class generates the exact URL request and then calls the **requestTokenizedURL()** method on your token service. Your token service class calls your token service and receives a token. If it is successful, it calls back the **callBackFunctionOnSuccess** function that the HDNetStream defined. If it has a problem, then it calls back the **callBackFunctionOnFailure** function.
8. Assuming success, the HDNetStream class hands the EdgeAuth token to the Edge server. If that token is validated, playback continues indefinitely. If the token is rejected, the Edge server disconnects the delivery session immediately.

   Token Authorization has this construct because authorization tokens might be required at any time during playback, not only at the beginning. This is because the Edge server has a short timeout period of only 120 seconds. If you pause the player for more than 120 seconds, the network connection times out. The HDNetStream detects this and makes a new **play()** request to continue playback. To do so, it needs to request fresh EdgeAuth tokens from your service.

How to use subtitles

Note: The Subtitles feature is a Beta version. Contact your Account Representative for access to this feature.

Subtitling is a process of overlaying video with text. This is done to provide accessibility to hearing-impaired people, language translation, or any other reason a producer chooses. Subtitles can be passed to a viewer in an accompanying *sidecar* file.

The following file formats supported for sidecar files:

- WebVTT — Web Video Text Tracks Format — Sidecar file format specified by W3C at *http://dev.w3.org/html5/webvtt/*.
- DFXP — Distribution Format Exchange Profile – Sidecar file format specified by W3C at *http://www.w3.org/TR/2006/CR-ttaf1-dfxp-20061116/*.

You need a separate sidecar file for each language.

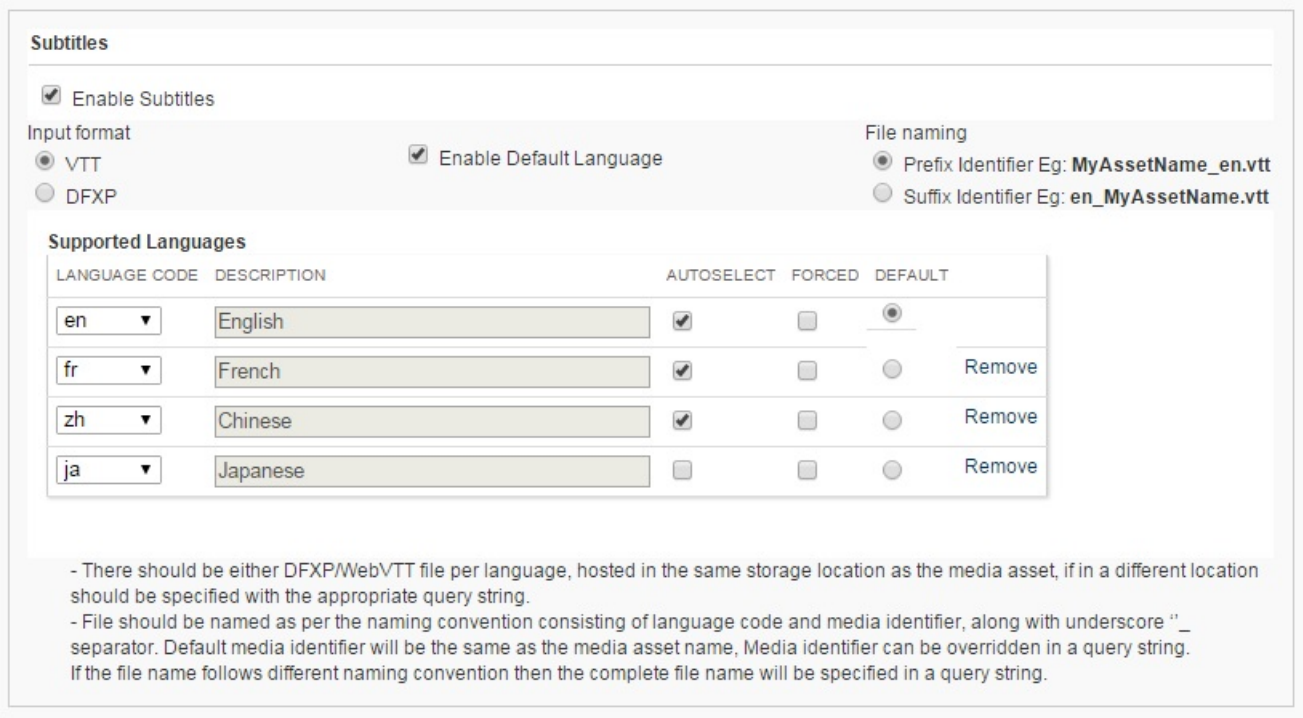
If multiple subtitles are configured, all subtitle files must be present or no subtitles will be displayed.

If you store your media assets and accompanying subtitles' files in the same location, you set up subtitles via the UI when creating your configuration. For information on subtitles embedded into the video file itself, see *Using closed captions* on page 29.

Examples of setting up Subtitles via UI

The Subtitles feature is a Beta version. Contact your Account Representative for access to this feature.

To illustrate how the enabled DEFAULT and AUTOSELECT options affect your playback, let us use the example shown in the following figure.



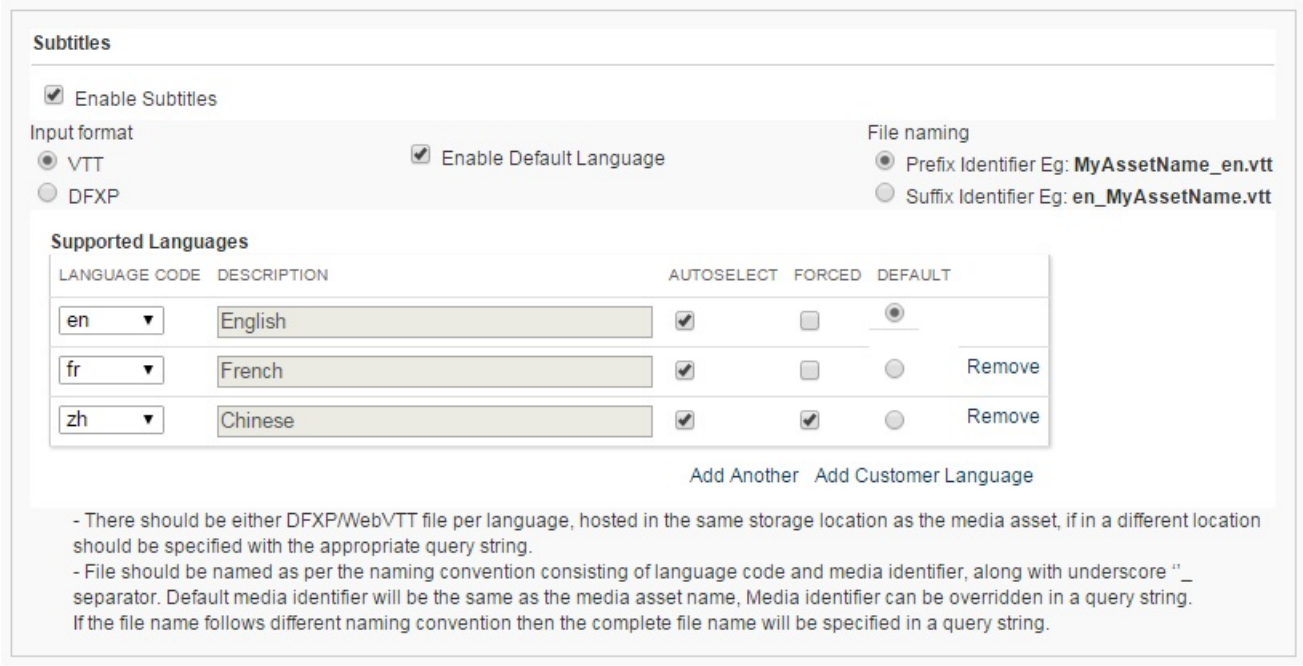
Example 1: Setting Up Subtitles During Provisioning on Control Center (Using AUTOSELECT and DEFAULT options)

In this example, our media asset for the movie could be accompanied by several subtitle files (English, French, Chinese, Japanese) that are preferably stored in the same location as the media asset.

- By DEFAULT, this movie is to be streamed with English subtitles.
- If the **Auto (Recommended)** option is selected during playback of this movie, the languages for subtitles are automatically selected from the list of the autoselected languages (out of English, French, and Chinese).

  For example, the player/device has a localization setting to French then a most appropriate language for the subtitles from your list of autoselected languages will be automatically chosen during playback — French. If this movie is streamed in China, the Chinese subtitles will be picked up, etc.

To illustrate how the enabled **FORCED** option affects your playback, let us use the example shown in the following figure.



Example 2: Setting Up Subtitles During Provisioning on Control Center (Using FORCED option)

In this example, even if the **CC** option is **Off** during playback, the **FORCED** subtitles will automatically be displayed — Chinese. The option to choose these Chinese **FORCED** subtitles will not be displayed during playback.

> **Note:** Some players do not have the **FORCED** option.

Query strings for setting up subtitles

You can use query strings to configure the subtitles for playback:

- If your media assets and accompanying subtitles' files are stored in different locations;
- To override the setup for Subtitles that is done via UI during creation of configuration.

subtitle_file_pattern

The `subtitle_file_pattern` query string can be one of the two following formats:

```
langcode delimiter identifier.extension
```

```
identifier delimiter langcode.extension
```

If subtitle_file_pattern is present

It denotes how the filename of a subtitle segment is constructed:

- *langcode* is a keyword that is replaced with the language codes of that stream.

- *delimiter* is the actual delimiter that is used by the customer.
- *identifier* is a keyword that is replaced either with *subtitle_identifier* or *media identifier*.

  Note: All the above 3 fields are required, if the subtitle_file_pattern query parameter is present.

- *extension* is an optional field that would have the actual extension that a customer would like to use. If *extension* is not present, the default extension **vtt** or **dfxp** is used (depending on input format).

If subtitle_file_pattern is absent

The file pattern prescribed in the configuration set up during provisioning on Control Center is used.

subtitle_identifier

The **subtitle_identifier** query string can be only a string.

If subtitle_identifier is present

The *identifier* in **subtitle_file_pattern** is replaced with this value.

If subtitle_identifier is absent

If absent, the media identifier is used to replace the *identifier* in **subtitle_file_pattern**.

subtitle_location

The **subtitle_location** query string can be only a string that present either an absolute or relative directory.

If subtitle_location is present

It denotes the alternate directory location where the subtitle segments are present.

If subtitle_location is absent

The subtitle segments are present in the same location as the media segments.

Query string examples for setting up subtitles

This section provides the examples of query string usage for setting up subtitles.

In all examples presented in the following table:

```
example_url = http://something-vh.akamaihd.net/i/dir1/dir2/
video_,100k,200k,500k,.mp4.csmil/master.m3u8
```

For <*example_url*>, the media files are:

```
cpcode/dir1/dir2/video_100k.mp4
```

```
cpcode/dir1/dir2/video_200k.mp4
```

```
cpcode/dir1/dir2/video_500k.mp4
```

For this example, *media identifier* is `video`. The underscore after `video` is not a part of the media identifier.

The subtitle `delimiter` in these examples is chosen to be an underscore.

The input subtitle format in these examples is assumed to be WebVTT. Hence, the default input subtitle file name extension is `.vtt`.

Query string examples for setting up subtitles

Use case	URL with example query parameters	Subtitle location	Subtitle identifier	Subtitle Filename
`subtitle_file_pattern=langcode_identifier`	*exampleurl*?subtitle_file_pattern=langcode_identifier	Same as media location. *cpcode*/dir1/dir2/	Same as media identifier `video`	`en_video.vtt` `zh_video.vtt`
`subtitle_file_pattern=identifier_langcode`	*exampleurl*?subtitle_file_pattern=identifier_langcode	Same as media location. *cpcode*/dir1/dir2/	Same as media identifier `video`	`video_en.vtt` `video_zh.vtt`
`subtitle_file_pattern=identifier_langcode.extension` `custom file extension = .abc`	*exampleurl*?subtitle_file_pattern=identifier_langcode.abc	Same as media location. *cpcode*/dir1/dir2/	Same as media identifier `video`	`video_en.abc` `video_zh.abc`
If `subtitle_file_pattern` is absent		Same as media location. *cpcode*/dir1/dir2/	Same as media identifier `video`	File pattern chosen during creating configuration on Control Center.
`subtitle_identifier=subtitles`	*exampleurl*?subtitle_file_pattern=identifier_langcode subtitle_identifier=subtitles	Same as media location. *cpcode*/dir1/dir2/	*subtitle_identifier* `subtitles`	`subtitles_en.vtt`
`subtitle_location=dir3` (Relative directory)	*exampleurl*?subtitle_file_pattern=identifier_langcode subtitle_identifier=subtitles&subtitle_location=dir3	New relative directory:	*subtitle_identifier* `subtitles`	`subtitles_en.vtt` under *cpcode*/`dir1/`

Use case	URL with example query parameters	Subtitle location	Subtitle identifier	Subtitle Filename
		cpcode/ dir1/dir2/ dir3		`dir2/ dir3`
`subtitle_location=/ dir3` (Absolute directory)	*exampleurl*? subtitle_file_pattern=identifier_langcode subtitle_identifier=subtitles &subtitle_location=/dir3	New absolute directory: *cpcode*/dir3	*subtitle_identifier* `subtitles`	`subtitles_ en.vtt` under *cpcode*/ `dir3`

Using closed captions

Closed captions is a process that allows insertion of data/text that corresponds to the video scene. This can be done to provide accessibility to hearing-impaired persons (close captions or open captions), language translations for multi-region programming, or any other reason a producer chooses. Closed captions can be passed to a viewer in one of these ways:

- Embedded into the video file itself, or
- Delivered in an accompanying “sidecar” file

Support for closed captions

Pre-generated “sidecar files” formatted as DXFP can be delivered via PMD alongside the HD video. The player has to do the work to fetch and synchronize the “sidecar” files. There are standard OSMF plugins that handle this functionality. HDCore has APIs support for the captioning/subtitling plugin as well. Refer to the sample under \samples\HDS\AkamaiDFXPCaptioningSample folder in the HDCore package.

Frequently asked questions

How do I allow only the highest bit rates when in full screen mode?

Assume you have a player that has a stage video of 640x360 and an MBR set of files at these rates:

- 640x360, 500 kbps
- 640x360, 1.5mbps
- 1280x720, 2.5mbps
- 1280x720, 3.5mbps

It would be a waste of bandwidth allowing the 3.5mbps feed to play in the 640x360 stage. The end user wouldn't notice a quality benefit and the player would have to spend CPU cycles throwing away most of the video content it was receiving.

1. To limit the player to use the 1280x720 renditions in fullscreen mode, set the **maximumBitrateAllowed** property to **1700** when the HDNetStream class is instantiated. (Any

number greater than 1500 will do.) This limits the class's switching algorithms to only allow a bit rate encoded at 1700 or below, and it will only cycle between the two lower renditions.

2. In your player, set a listener on the fullscreen event, and once the player is in fullscreen mode, set **maximumBitrateAllowed** to any number greater than 3500. The class can switch up to the HD resolutions if the user has sufficient bandwidth.

3. Once the user exits fullscreen mode, then you could set **maximumBitrateAllowed** back to 1700. The class switches down to the highest bit rate it can sustain that is lower than that value, effectively limiting the user to a 640x360 video.

How do I determine which index the class uses when it starts a rendition set?

By default, the HDNetStream class starts a rendition set at the highest bit rate, that is less than 500 kbps. If you want to modify this behavior, you have two options:

- Before calling **play()**, set the **startingIndex** property to the index value you want to use. Index values are zero-based, with the renditions ordered by ascending bit rate.

- For a more sophisticated entry point selection, extend HDNetStream and then override the protected**getStartingIndex()** function.

How can I debug what is happening within the HDNetStream class?

The class dispatches an event called **DEBUG**. This contains a data string that traces the internal activity of the class. This is extremely useful information to provide to Media Services On Demand: Stream Packaging when debugging. It is highly recommended you create a mode in your player allowing you to listen to and trace out these debug messages, or perhaps add them to your existing logger.

If I view HD headers, I see many requests being made during playback. Why is that?

Each HDNetStream class will set up two concurrent HD sessions to the edge server. The first, called the delivery session, is the one that delivers the actual video content. This stays connected as long as the video is being downloaded and played. The second session, called the control session, is used by the class to communicate with the server. This communication channel allows it to control the buffer length on the delivery session and also to switch the content that is playing on the delivery session to another bit rate.

What is the timeout value on the Media Services On Demand edge servers?

This is currently two minutes. If you pause the stream for more than 120 seconds, the server times out. If you press play again, the class plays whatever buffer it had built up and then receives a **TIMEOUT** marker from the server. At this point, it automatically reconnects and resumes playback at the point before the timeout.

Set up the video playback for HLS output

Unlike HDS and HD Flash 1.0. playback, playback for HLS streams is accomplished with end users' native media players. So, no direct action on the playback client is required on your part. The following sections will be helpful to you in setting up other aspects of your streams' playback.

Helpful definitions for HLS output

To facilitate your use of this user guide and Media Services On Demand: Stream Packaging itself, this section introduces you to some key terms and definitions you may find useful.

File Type	Description	Extension	Example
Variant Playlist	This file, which the player requests just once at the start of the session, contains a list of all the requested stream's available bit rates (tracks). Since the file includes each track's bandwidth, the player uses this bandwidth information to switch between the most appropriate bit rates in case of network bandwidth fluctuations.	m3u8	master.m3u8
Child Playlist	Each track has its own child playlist file that contains a list of segments available for playback. The number of segments in the file indicates duration of the content.	m3u8	index_1000kpbs.m3u8
Segment	This file contains the video and/or audio data in MPEG2-TS format. For audio-only tracks, the data is contained in AAC format with an .aac extension.	ts / aac	segment10.ts
Encryption Key	This file includes a 128-bit key used to encrypt the segments. It is only requested when encryption is turned **on** at the encoder.	key	crypt.key

Using other compatible playback methods

- HLS Protocol, Version 2 and above

  Refer to *IETF Internet Draft of the HD Live Streaming Protocol Specification*.

- iOS, Version 3 and above
- HTML5

  Supports the *video* tag in the Apple Safari® browser on the Apple OS X® operating system.

- Apple QuickTime® application program on Apple Mac OS® operating system and iOS devices

Using Media Services On Demand support players

You can use the provided support video player to test and troubleshoot your streams: *iDevices Support Player*

Serving your content

Any device that supports HLS output will directly support a segmented streaming approach and launches when you enter an M3U8/M3U link into the Apple Safari® browser. Publishers can also choose to write their own native iPhone applications that are able to seamlessly browse content, leveraging the Media Player for content rendering.

Using SMIL files

This is the usual form for passing your streams' information to the end-user client. For Apple HTTP Streaming, the SMIL publish URL takes the form:

```
http://customer-vh.akamaihd.net/i/smil_file_name.smil/master.m3u8
```

So an actual SMIL URL might be something along the lines of:

```
http://example-vh.akamaihd.net/i/example2a.smil/master.m3u8
```

Following is an example of the format for the contents of SMIL file—example2a.smil:

```
<?xml version="1.0"?>
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN" "http://www.w3.org/2001/
SMIL20/SMIL20.dtd">
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
<head> <meta name="title" content="Event" />
<meta name="httpBase" content="http://example.akamaihd.net/" />
</head>
<body>
<switch id="whatever">
<video src="subdir/example2a_300000.mp4" system-bitrate="300000"/>
<video src="subdir/example2a_500000.mp4" system-bitrate="500000"/>
<video src="subdir/example2a_800000.mp4" system-bitrate="800000"/>
<video src="subdir/example2a_1000000.mp4" system-bitrate="1000000"/>
</switch>
</body>
</smil>
```

Using the client-side URL syntax

If you are unable to generate or provide server-side SMIL files, you can alternatively use a client-side URL format with which you can encode the entire SMIL contents in a single URL. For HLS, the client-side URL takes the form:

```
http://customer-vh.akamaihd.net/i/subdirectory/common_filename_prefix,
bitrate,bitrate,bitrate,bitrate,
common_filename_suffix.csmil/master.m3u8
```

So a published URL to the device for this form of integration might look like:

```
http://example-vh.akamaihd.net/i/movies/example2a_,
300000,500000,800000,1000000,_event1.mp4.csmil/master.m3u8
```

Using the above examples and assuming a CP code of 9389, your content would reside on NetStorage at `example.download.akamai.com/9389/movies/` as:

- `example2a_300000_event1.mp4`
- `example2a_500000_event1.mp4`
- `example2a_800000_event1.mp4`
- `example2a_1000000_event1.mp4`

> **Note:** If you plan to stream just a single bitrate file, it does not need client-side URL and could be added by itself (that is a single mp4 file called hello.mp4 could have a syntax of *hostname*`/i/hello.mp4/master.m3u8`).

HLS delivery improvements per HLS spec upgrade

As part of HLS delivery improvements, a new query parameter `set-akamai-hls-revision` has been introduced to enable HLS manifest improvements and HLS specification upgrades. This query parameter specifies the HLS revision

> **Note:** `set-akamai-hls-revision` number does not correspond to the HLS Spec Upgrade version number. It is an internal company setting that results in the following:

- If `set-akamai-hls-revision` is set to **4**, the HLS playlist version (EXT-X-VERSION) is **3**;
- If `set-akamai-hls-revision` is set to **5**, the HLS playlist version (EXT-X-VERSION) is **4**.

When you specify the `set-akamai-hls-revision` query string, the following features are automatically enabled as listed in the following table.

Features enabled with revision 4 and 5

Feature	Playlist tag name
If `set-akamai-hls-revision` is set to **4** or **5**, these features are enabled:	
Qualify to enable closed captions control option in stream-inc	#EXT-X-STREAM-INF:<attribute-list>
Remove PROGRAM-ID attribute from EXT-X-STREAM-INF tag	#EXT-X-STREAM-INF:<attribute-list>
Put AVERAGE-BANDWIDTH attribute to EXT-X-STREAM-INF tag	#EXT-X-STREAM-INF:<attribute-list>
If `set-akamai-hls-revision` is set to 5, this additional feature is also enabled:	

Feature	Playlist tag name
The CODEC parameters that contain the profile and level values for H.264 streams would be advertised in conformance to RFC 6381.	#EXT-X-VERSION:<version> #EXT-X-STREAM-INF:...,CODECS=<codecs>

Note: If a decision is made to advertise independent segments that are key frame aligned, care must be taken to ensure that the source content has been encoded with key frames at intervals that are “factors of the segment duration”. For example, if the chosen segment duration is 10 seconds, then the source content should have been generated with key frames at 1, 2, or 5-second intervals (the GOP size should be 1, 2, or 5 seconds).

The following example shows a playback with the set-akamai-hls-revision query string set to 5.

```
http://example-vh.akamaihd.net/i/xyz.mp4/index_0_a.m3u8?set-akamai-hls-revision=5
```

Using query strings

There are several query strings available for use with your client-side URL for HLS output:

Note: The values below are in kbps and are applied to the combined audio + video bit rate.

Bandwidth correction (bwcorrection)

This query string can be used to adjust the bitrate advertised in the Variant Playlist’s BANDWIDTH attribute (values are space-delimited). Values can be from 0 to +/-100.

```
uri/master.m3u8?bwcorrection=5 10-10
```

Note: When used, this will be the last filter applied.

Bit rate filtering (b)

If you would like to filter out certain bit rates—to test a particular bit rate version, for example—you can do so by adding a `b` query string to the end of the publish URL. There are four uses:

- Filter all bit rates except those specifically designated (and the lowest audio rate); the string takes the form:

  ```
  b=bitrate,bitrate,bitrate,...
  ```

  So, for example, if you had bit rates 200, 300, 700, and 1200, and you wanted to include only bit rates 200, 700, and the lowest audio rate, your URL would appear as:

  ```
  uri/master.m3u8?b=200,700
  ```

This would produce only the 200 and 700 bit rate streams, as well as the lowest audio bit rate.

- Designate a bit rate range (in kbps) outside of which the stream or streams you wish to filter fall; the string takes the form:

```
b=lower_end-upper_end
```

So, for example, if you had bit rates 200, 300, 700, and 1200, and you wanted to include all bit rates at or greater than 200 and at or less than 700, your URL would appear as:

```
uri/master.m3u8?b=200-700
```

This would produce the 200, 300, and 700 bit rate streams.

- Designate the low end of a bit rate range (in kbps) where all bit rates below the value are filtered; the string takes the form:

```
b=low_end-
```

So, for example, if you had bit rates 200, 300, 700, and 1200, and you wanted to include all bit rates at and higher than 300 kbps, your URL would appear as:

```
uri/master.m3u8?b=300-
```

This would produce the 300, 700, and 1200 bit rate streams.

- Designate the high end of a bit rate range (in kbps) where all bit rates at and above the designation are filtered; the string takes the form:

```
b=0-high_end
```

So, for example, if you had bit rates 200, 300, 700, and 1200, and you wanted to include all bit rates lower than or equal to 700 kbps, your URL would appear as:

```
uri/master.m3u8?b=0-700
```

This would produce the 200, 300, and 700 bit rate streams.

- Any combination of the above.

  So, for example, if you had bit rates 200, 300, 700, 1200, and 1800, and you wanted to include all bit rates except 700, your URL could appear as:

```
uri/master.m3u8?b=0-300,1200-
```

This would produce only the 200, 300, 1200, and 1800 bit rate streams, as well as the lowest audio bit rate.

> **Note:** If Bit Rate Filtering and the Extracted Audio-Only features are simultaneously enabled, bit rate filtering is applied first, then audio-only extraction is applied to the remaining bit rates.

Extracted audio-only stream (__a__)

When enabled in a Media Services One Demand (Stream Packaging) configuration, the Edge server automatically generates an audio-only stream from one of your bit rate streams. You may use the `__a__` query string to disable this feature on a per-stream basis, if desired. Available values are ‘on” and “off.” For example:

```
uri/master.m3u8?__a__=off
```

Note: This query string is only effective if you are using the extracted audio-only stream. If Bit Rate Filtering and the Extracted Audio-Only features are simultaneously enabled, bit rate filtering is applied first, then audio-only extraction is applied to the remaining bit rates.

Flexible playback (start | end)

This query string enables end-users to begin playback at a specific time (in seconds) within the VoD stream. For example:

```
uri/master.m3u8?start=30&end=500
```

Note: The start time is mandatory but the end time is not. If the end time is not provided, the stream will play until the end of the VoD asset.

Master playlist optional attribute removal (attributes)

Each bitrate in the Master playlist is represented by different attributes, for instance, BANDWIDTH, CODECS, and RESOLUTION. Some of these attributes are optional as per HLS specification. This query string when set to **off** will remove the optional attributes. By default, this option is **on**.

Note: It has been observed that certain devices and platform, like XBOX 360, do not ignore these optional tags if the implementation does not support them. This behavior is not specification complaint. Use of attributes=off query string would let you play the content on these devices by removing the optional attributes in this file.

For example, applying this string query to the following attributes highlighted in bold:

```
#EXTM3U
#EXT-X-STREAM-INF:PROGRAM-
ID=1,BANDWIDTH=1128000,RESOLUTION=960x540,CODECS="avc1.77.30, mp4a.40.2"
http://example-vh.akamaihd.net/i/movies/event.smil/index_1000_av-p.m3u8?sd=10
#EXT-X-STREAM-INF:PROGRAM-
ID=1,BANDWIDTH=2528000,RESOLUTION=1280x720,CODECS="avc1.77.30, mp4a.40.2"
http://example-vh.akamaihd.net/i/movies/event.smil/index_2400_av-p.m3u8?sd=10
#EXT-X-STREAM-INF:PROGRAM-
ID=1,BANDWIDTH=3128000,RESOLUTION=1280x720,CODECS="avc1.77.30, mp4a.40.2"
http://example-vh.akamaihd.net/i/movies/event.smil/index_3000_av-p.m3u8?sd=10
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=139000,CODECS="mp4a.40.2"
http://example-vh.akamaihd.net/i/movies/event.smil/index_1000_a-p.m3u8?sd=10
```

will result in the following:

```
#EXTM3U
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1128000
http://example-vh.akamaihd.net/i/movies/event.smil/index_1000_av-p.m3u8?sd=10
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=2528000
http://example-vh.akamaihd.net/i/movies/event.smil/index_2400_av-p.m3u8?sd=10
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=3128000
http://example-vh.akamaihd.net/i/movies/event.smil/index_3000_av-p.m3u8?sd=10
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=139000
http://example-vh.akamaihd.net/i/movies/event.smil/index_1000_a-p.m3u8?sd=10
```

Preferred bit rate (__b__)

If desired, you can designate a specific bit rate stream to be the “preferred” bit rate, meaning the end user’s player would start with this bit rate before adjusting to another bit rate depending on network conditions. For example:

```
uri/master.m3u8?__b__=500
```

Note: This query string is ineffective with Microsoft® Xbox® video player, as it automatically selects the lowest bit rate at the beginning.

Testing your content

You should test your content prior to making it available to your end-users to help ensure they will have a problem-free experience.

Note: Staging hostnames are added automatically for all **akamaihd.net** domains in order to test configurations on the Staging network. For example, if your akamaihd hostname is **example-vh.akamaihd.net**, your staging hostname will be example-vh.akamaihd-staging.net.

How to

1. Compose your URLs and request the .m3u8 and segment files directly from a Web browser to check if they are cached and of the correct duration. This will confirm whether the connection delivery is correct. Be aware, you will not see video playback during this test.
2. Using the iPhone emulator, iPhone, iPad, or iPod Touch® with the latest iOS version, request the file to watch it and ensure it plays correctly.

Configure security

Security for HLS output is provided by the SecureHD Policy Editor (SPE), which is configurable through Control Center. For information regarding SPE and its configuration, refer to *Media Security Policy Editor User Guide* also available in Control Center.

Note:

- SPE's Player Verification does not currently support HLS output. Also, Encryption Percentage, a part of SPE's Media Encryption feature, does not apply to Apple HTTP Streaming.
- SecureHD Media Encryption feature is not supported for HLS with Additional Digital Properties, meaning, the hostname does not end with akamaihd.net or akamaihd-staging.net domain
- It is recommended that you do not have more than 10 hostnames associated with SecureHD Policy Editor for performance reasons. If you require more than 10 hostnames associated with SecureHD Policy Editor, contact your Account Representative for assistance.

Using closed captions

Closed captions is a process that allows insertion of data/text that corresponds to the video scene. This can be done to provide accessibility to hearing-impaired persons (close captions or open captions), language translations for multi-region programming, or any other reason a producer chooses. Closed captions can be passed to a viewer in one of these ways:

- Embedded into the video file itself, or
- Delivered in an accompanying "sidecar" file

Support for closed captions

In the h.264 input stream, if the subtitles text is present in the NALU header type 6 (SEI) as ATSC picture user data, it will be passed through and available in the HLS output.

Purging HDS and HD Flash 1.0 video content

> **Note:** The best practice is always to upload the updated content under a new name and then update your URL accordingly.

However, there might be situations in which you will want to replace new content while using the same file name. This would involve uploading the new content in place of the old content.

However, edge servers might retain the old content in their caches, requiring you to purge those caches to allow end-users to play the new content.

If you do decide to use in-place content purge, the next section describes the procedures. Also, note that in-place cache purging can be time intensive (approximately 90 minutes).

> **Note:** Because of the nature of the multiple bitrate feature, you must purge every version of the content in question. Failure to do so can result in unpredictable behavior.

There are different procedures for purging content from the streaming cache depending on what formats you are using: HDS or HD Flash 1.0.

Purging streaming cache of HDS content

To illustrate the procedures below, the following assumptions are used as an example.

The manifest URL:

```
http://example-vh.akamaihd.net/z/example/event1.smil/manifest.f4m
```

And these files residing on the origin:

- event1_220K_video.mp4
- event1_500K_video.mp4
- event1.smil

> **Note:** Be certain you use the provided hostname and not your own CNAME'd hostname in the purge URL.

Use Control Center's HD Content Control Utility to purge your Flash media.

How to

1. Replace the content in question on your origin.
2. If you are using NetStorage for your origin, refer to *NetStorage User Guide* for information regarding accessing your account.

3. Log in to Control Center.
4. In the upper navigation bar, click the **PUBLISH** tab. The **Publish** menu appears.
5. Select **Content Control Utility**. The **Select Product** page appears.
6. Select the **HD On Demand Streaming** radio button and click **Continue**. The **HD Content Control Utility** page appears.
7. Select the **HD On Demand Streaming** radio button and click **Continue**. The **HD Content Control Utility** page appears.
8. From the **Content Type** drop-down menu, select **Adobe Dynamic Streaming**.
9. If desired, enter a name for the request in the **Request Name (Optional)** text box.
10. In the **Content to Refresh** area:

If...	Then...
You want to purge by URL/ARL	1. Select the URL/ARL entered below radio button. 2. In the first text box, enter the manifest URL from which content is to be purged. For example: `http://example-vh.akamaihd.net/z/examplehost/event1.smil/manifest.f4m` 3. In the second text box, enter the file names to be purged, separating each with a newline. For example: • SMIL: event1_220K_video.mp4, event1_500K_video.mp4, event1.smil • Single File: event1_220K_video.mp4 • Client-Side URL Syntax: example2a_,300000,500000,800000, 1000000,_event1.mp4.csmil • Manifest: Specify the manifest (.f4m), index (.f4x), and data files (.f4f) – Manifest: For example, event1.f4m – Index: For example, event1_220K_videoSeg1.f4x – Data: For example, event1_220K_videoSeg1.f4f
You want to purge all URLs/ ARLs.associated with specific CP codes	1. Select the Refresh ALL URLs/ARLs Associated with Specific CP Codes radio button. 2. Select the check box of any CP codes for which you would like to purge content.

If...	Then...
Note: Use care when using this feature.	

11. In the **Notification** area, select whether you would like to receive an e-mail notification when the content purge is complete.

    If you choose to receive notification, enter your e-mail address in the accompanying text box.

12. Click **Start Refreshing Content**.

What you should see

The purge is initiated. An e-mail will be forthcoming if you opted to for email notification to signal the completion of the process.

Note: Be aware that cache purges can take approximately 90 minutes to take effect.

Purging the streaming cache of HD Flash 1.0 content

Before you begin

Before purging the streaming cache of HD Flash 1.0, you must construct a purge URL for your content.

The purge URL is fairly simple to construct, simply being the hostname, path, and file name of the content you wish to remove in the form:

```
http://Akamai_hostname/directory/directory/.../content_name.
```

Note: Be certain you use the provided hostname and not your own CNAME'd hostname in the purge URL.

An example of a purge URL might be:

```
http://examplehost-vh.akamaihd.net/movies/trailers/trailer1.mp4
```

This is the URL you will use in the HD CCU.

How to

1. Access the HD Content Control Utility:

    a. Log in to Control Center.

    b. In the upper navigation bar, click the **PUBLISH** tab.
    The **Publish** menu appears.

    c. Select **Content Control Utility**.

The **Select Product** page appears

d. Select either the **Media Services On Demand** radio button and click **Continue**.
The **HD Content Control Utility** page appears.

2. Purge the Flash video from the streaming cache.

a. From the **Content Type** drop down menu, select one of the following:

- HD Flash 1.0
- HLS
- HDS

b. In the **Content to Refresh** area, enter your purge URL in the box; separate multiple purge URLs with a new line.

c. In the **Notification** area, select the desired radio button to either receive or not receive an e-mail notification upon removal of the content. If you choose to receive notification, enter your e-mail address in the text box.

d. Scroll to the bottom of the page and click **Start Refreshing Content**.The content will be purged from Media Services On Demand infrastructure. Afterward, the next end-user request for the video will cause the service to download the new content from your Origin location.

ⓘ **Note:** Be aware that cache purges can take half an hour or longer to take effect.

Purging HLS video content

Note: The best practice is always to upload the updated content under a new name and then update your URL accordingly.

However, there might be situations in which you will want to replace new content while using the same file name. This would involve uploading the new content in place of the old content.

However, Edge servers might retain the old content in their caches, requiring you to purge those caches to allow end-users to play the new content.

If you do decide to use in-place content purge, the next section describes the procedures. Also, note that in-place cache purging can be time intensive (approximately 90 minutes).

Note: Because of the nature of the multiple bitrate feature, you must purge every version of the content in question. Failure to do so can result in unpredictable behavior.

To illustrate the procedures below, the following assumptions are used as an example.

The manifest URL:

```
http://example-vh.akamaihd.net/i/examplehost/event1.smil/master.m3u8
```

These files are residing on your origin:

- event1_220K_video.mp4
- event1_500K_video.mp4
- event1.smil

Note: Be certain you use the Media Services On Demand: Stream Packaging hostname and not your own CNAME'd hostname in the purge URL.

Use Control Center's HD Content Control Utility to purge your media.

How to

1. Replace the content in question on your origin.
2. If you are using NetStorage for your origin, refer to *NetStorage User Guide* for information regarding accessing your account.
3. Log in to Control Center.
4. In the upper navigation bar, click the **PUBLISH** tab. The **Publish** menu appears.
5. Select **Content Control Utility**. The **Select Product** page appears.
6. Select the **HD On Demand Streaming** radio button and click **Continue**. The **HD Content Control Utility** page appears.

7. From the **Content Type** drop down menu, select **HD Apple iPhone/iPad On Demand**.
8. If desired, enter a name for the request in the **Request Name (Optional)** text box.
9. In the **Content to Refresh** area:

If...	Then...
You want to purge by URL/ARL	1. Select the **URL/ARL entered below** radio button. 2. In the first text box, enter the manifest URL from which content is to be purged, for example: `http://example-vh.akamaihd.net/examplehost/event1.smil/master.m3u8` 3. In the second text box, enter the file names to be purged, separating each with a newline. For example: • SMIL: event1_220K_video.mp4, event1_500K_video.mp4, event1.smil • Single File: event1_220K_video.mp4 • Client-Side URL Syntax: example2a_,300000,500000,800000, 1000000,_event1.mp4.csmil
If you want to purge all URLs/ARLs associated with specific CP codes ⓘ **Note:** Use care when using this feature.	1. Select the **Refresh ALL URLs/ARLs Associated with Specific CP Codes** radio button. 2. Select the check box of any CP codes for which you would like to purge content.

10. In the **Notification** area, select whether or not you would like to receive an e-mail notification when the content purge is complete.
    If you choose to receive notification, enter your e-mail address in the accompanying text box.
11. Click **Start Refreshing Content**.
    The purge is initiated. An e-mail will be forthcoming if you opted to for e-mail notification to signal the completion of the process.

ⓘ **Note:** Be aware that cache purges can take approximately 90 minutes to take effect.

Reporting

Several historical data reports are available to you via Akamai Control Center.

These include traffic reports, visitor reports, URL reports, and URL locations reports. You view various data about your MSOD instance and you can customize your reports. Usage for *HDS/HLS* and *HD Flash 1.0* content reports are covered in the *Media Services Reports User Guide*.

Notice

Akamai secures and delivers digital experiences for the world's largest companies. Akamai's Intelligent Edge Platform surrounds everything, from the enterprise to the cloud, so customers and their businesses can be fast, smart, and secure. Top brands globally rely on Akamai to help them realize competitive advantage through agile solutions that extend the power of their multi-cloud architectures. Akamai keeps decisions, apps, and experiences closer to users than anyone — and attacks and threats far away. Akamai's portfolio of edge security, web and mobile performance, enterprise access, and video delivery solutions is supported by unmatched customer service, analytics, and 24/7/365 monitoring. To learn why the world's top brands trust Akamai, visit *www.akamai.com*, *blogs.akamai.com*, or *@Akamai* on Twitter. You can find our global contact information at *www.akamai.com/locations*.

Akamai is headquartered in Cambridge, Massachusetts in the United States with operations in more than 57 offices around the world. Our services and renowned customer care are designed to enable businesses to provide an unparalleled Internet experience for their customers worldwide. Addresses, phone numbers, and contact information for all locations are listed on *www.akamai.com/locations*.

Published 1/2022