

EdgeCast Networks, Inc.

HTTP Streaming



Disclaimer

Care was taken in the creation of this guide. However, EdgeCast Networks, Inc. cannot accept any responsibility for errors or omissions. There are no warranties, expressed or implied, including the warranty of merchantability or fitness for a particular purpose, accompanying this product.

Trademark Information

EDGECAST is a registered trademark of EdgeCast Networks, Inc.

FLASH and AIR are registered trademarks of Adobe Systems Incorporated.

JW PLAYER is a registered trademark of LongTail Video.

FLOWPLAYER is a registered trademark of Flowplayer Ltd.

WINDOWS and SILVERLIGHT are registered trademarks of Microsoft Corporation.

IOS, IPAD, APPLE TV, and SAFARI are registered trademarks of Apple Inc.

About This Guide

HTTP Streaming Administration Guide

Version 1.66

3/31/2014

© EdgeCast Networks, Inc. All rights reserved.

Table of Contents

HTTP Streaming Solutions.....	1
Solution Comparison.....	1
Reports & Analytics.....	2
HTTP Progressive Download.....	3
Introduction.....	3
HTTP Progressive Download Requirements.....	3
Setting Up HTTP Progressive Download.....	4
Media Player.....	4
Sample Video Player Code.....	5
Seeking Within a Video.....	5
Providing Seek Support for a Custom Flash Player.....	8
HTTP Live Streaming.....	9
Introduction.....	9
HTTP Live Streaming Requirements.....	9
How Does It Work?.....	11
Live Streaming.....	11
On-Demand Streaming.....	11
Setting up HTTP Live Streaming.....	12
Live Streaming.....	12
On-Demand Streaming.....	22
Stream Encryption.....	25
PHLS Configuration (Live Streaming).....	26
PHLS Configuration (On-Demand Streaming).....	26
Purging Live & On-Demand Content.....	27
Purging a Live Event Example.....	29
Purging On-Demand Content Example.....	29

HTTP Dynamic Streaming.....	30
Introduction	30
HTTP Dynamic Streaming Requirements	30
How Does It Work?	32
Live Streaming.....	32
On-Demand Streaming	32
Setting up HTTP Dynamic Streaming	33
Live Streaming.....	33
On-Demand Streaming	42
Purging Live & On-Demand Content.....	46
Purging a Live Event Example	47
Purging On-Demand Content Example	47












HTTP Streaming Solutions

Solution Comparison

We provide various solutions through which you can stream live and on-demand content over the HTTP Large platform. Each streaming solution supports a different feature set. A brief description is provided below for major streaming features.

- **Live Streaming:** Indicates the capacity to stream a live event as it occurs.
- **On-Demand Streaming:** Indicates the capacity to stream media as it is requested.
- **Progressive Streaming:** Indicates that a media player will request media in a progressive sequence from start to finish. Normally, this would prevent a user from jumping ahead (i.e., seeking) to a position in the video that has not yet been downloaded. However, our technology allows this capability under certain circumstances. For more information, please refer to the **Seeking within a Video** section of the **HTTP Progressive Download** chapter.
- **Dynamic Streaming:** Indicates that it has the capability to broadcast multiple streams of varying quality. This allows each client's media player to intelligently and dynamically choose the stream that best matches the client's available bandwidth and CPU processing power.
- **Supported Media Player(s):** Indicates the media players that are natively supported by the streaming solution. Keep in mind that this field is not meant to be exhaustive.

The following table indicates the features/players that each of these solutions natively support.

Name	Live Streaming	On-Demand Streaming	Progressive Streaming	Dynamic Streaming	Supported Media Players
HTTP Progressive Download					All*
HTTP Live Streaming					iOS 3.0+; Safari 4.0+; Limited Android 4.0/4.1+ Support*
HTTP Dynamic Streaming					Adobe Flash Player 10.1+; Adobe AIR 2+
Smooth Streaming					Microsoft Silverlight 2+; OVP Silverlight; iOS/Safari*

Note: The HTTP Progressive Download streaming solution can be used with any media player that supports the requested media format (e.g., flv, mp4, mov, etc.).

Note: Android 4.0 and 4.1+ provide limited native support for HTTP Live Streaming. [View a sampling of the issues](#) that your viewers may encounter.

Note: Smooth Streaming only supports iOS and Safari when streaming a live event. For more information on Smooth Streaming, please refer to the **Smooth Streaming Administration Guide**.

Reports & Analytics

CDN activity for our HTTP Progressive Download, HTTP Live Streaming, HTTP Dynamic Streaming, and Smooth Streaming solutions is tracked under the HTTP Large platform. Each module in the EC360 Analytics Suite allows you to generate reports on the HTTP Large platform. These reports will contain data for these streaming solutions and standard traffic on the HTTP Large platform.

HTTP Progressive Download

Introduction

HTTP Progressive Download uses the HTTP protocol to stream archived media content. As a result, it cannot take advantage of the security features that are available with the RTMP protocol, such as encryption and SWF verification. Additionally, the entire video is downloaded to a user's computer. As a result of all of these factors, HTTP Progressive Download is more susceptible to piracy than some of our other streaming solutions.

As previously mentioned, the HTTP Progressive Download uses the HTTP protocol to stream video on-demand. Specifically, HTTP Progressive Download should only be used with the HTTP Large platform. This allows progressively downloaded content to take advantage of all of the features provided by the HTTP Large platform. For example, you can use an existing customer origin to indicate the location of your video content.

Note: Any type of asset supported by your video player can be streamed through HTTP Progressive Download. However, seeking within a video is only supported for FLV and H.264 encoded video.

HTTP Progressive Download Requirements

HTTP Progressive Download requires an origin server, our content delivery network, and a video player. The requirements for each of these items are listed below.

System Requirements:

- **Storage:** CDN Storage (i.e., CDN Origin Server) or Customer Origin Server
- **Content Delivery Network:** We provide servers that transmit your on-demand content to all necessary edge servers for delivery to your end-users.
- **Video Player:** Make sure that your video player supports the type of media that you would like to stream. No additional requirements are needed for HTTP Progressive Download.

Setting Up HTTP Progressive Download

Setting up HTTP Progressive Download involves the following steps:

1. Upload your media content to either CDN or a customer origin server.
2. Point your media player to the appropriate player URL. Make sure to use either a CDN or edge CNAME URL.

Note: For additional information on how to setup an origin server and upload content, please refer to the **Configuring an Origin Server** chapter.

Media Player

Once the desired media content is hosted on a customer or CDN origin server, you are ready to set up your media player for use with HTTP Progressive Download. This requires that you set the player URL to a CDN or edge CNAME URL that leverages the HTTP Large platform (as indicated below).

- **CDN URL:** `http://wpc.xxxx.edgecastcdn.net/yyxxxx/Path/Filename`
- **Edge CNAME URL:** `http://cname.hostname.com/Path/Filename`

When defining a CDN URL, you should keep the following items in mind:

- **Account number:** The term `xxxx` should be replaced with your MCC account number. This account number can be found in the upper-right hand corner of the MCC.
- **Origin identifier:** The term `yy` should be replaced with one of the following:
 - **Customer origin server:** It should be replaced with "80" (e.g., `/800001`).
 - **CDN origin server:** It should be replaced with "00" (e.g., `/000001`).
- **Path:** The term *Path* should be replaced by the path to the folder where the desired media content is stored.
 - **Customer origin server:** The specified path should start with the name of the customer origin configuration that points to your origin server.
 - **CDN origin server:** The entire path to the desired asset can be copied from the FTP client provided in the MCC. Simply click on the **Storage** tab, navigate to the desired folder, select the desired asset, make sure that the **HTTP Large** option has been selected, and then copy the URL that appears directly to the left of it.
- **Filename:** The term *Filename* should be replaced by the filename of the desired media asset (e.g., `video.flv`).
- **Token-Based Authentication:** If the desired video has been secured by Token-Based Authentication, then you will need to append to the filename a question mark (?)

followed by a token value that corresponds to an encryption key. For example, if the filename is "video.mp4" and the desired token is "a4fbc3710fd3449a7c99984d1b86603c22be1006d830b," then you would specify "video.mp4?a4fbc3710fd3449a7c99984d1b86603c22be1006d830b" as your filename. For more information, please refer to the **Token-Based Authentication User Guide**, which is available from the Media Control Center (MCC).

Tip: For more information on edge CNAME URLs, please refer to the **Masking the URL of a CDN Origin Server (CNAME)** section below.

Sample Video Player Code

Sample JW player code is provided in the following article:

- [JW Player 6 Sample Code for HTTP Progressive Download](#)

Seeking Within a Video

HTTP Progressive Download provides seeking functionality for FLV and H.264 encoded videos (i.e., MP4, F4V, and MOV). This functionality allows clients to seek to a particular position in the video, regardless of whether the video has already been cached by the user agent (i.e., web browser) that requested it. Additionally, it allows you to programmatically start a video at a particular point in time.

Important: If you would like to seek within an FLV asset, then it must be encoded with an OnMetaTag event as well as key frames.

Important: If you would like to seek within an H.264 encoded (e.g., MP4) asset, then the edge server must have the entire asset in cache, or an H.264 streaming module must be installed on the origin server and it must be configured to support ec_seek for H.264 encoded assets. The ec_seek functionality has been properly configured on our CDN origin servers for MP4 assets. For all other H.264 file formats, please contact your CDN account manager to activate ec_seek functionality on a CDN origin server.

Important: Keep in mind that unique query string caching will cache each seek request. It is recommended that you limit unique query string caching to those folders where dynamic content is generated. By excluding folders that contain media content, your clients will be able to seek without caching multiple versions of the same asset. For more information on how to customize your unique query string caching configuration, please contact your CDN account manager.

A video player can be implemented with a scrubber bar (a.k.a. seek bar). A user can jump to a particular position in the video by clicking on the desired position in the scrubber bar. However, before you can take advantage of this functionality you will need to provide a hint to your video player to use "ec_seek" instead of its native parameter. Below you will find an example of how to implement this configuration for JW Player.

- **JW Player 5.x (5.3 and above):** 'http.startparam': 'ec_seek'
- **JW Player 5.1:** http.startparam=ec_seek

Note: For information on how to configure JW Player 5.x, please refer to the documentation that was provided with it.

Note: The "ec_seek" parameter is unnecessary when using JW Player 6.

Note: If you would like to configure a different video player, keep in mind that the seek parameter (e.g., http.startparam) may be different. Please refer to that documentation provided with the desired player.

Once you have configured your video player to use the "ec_seek" parameter, a user can jump ahead to portions of the video that have not yet been downloaded to his/her computer. This can be performed using a player-specific seeking parameter. For example, JW Player uses "start" as can be seen in the following code fragment:

```
file: "http://wpc.0001.edgecastcdn.net/000001/Video01.mp4",
'http.startparam': 'ec_seek',
start: 30
```

Note: The above sample code assumes that the requested video was encoded using H.264 format. It will seek 30 seconds into an H.264 encoded video.

Note: The above scenario assumes that a user chose to skip ahead to a portion of the video that had not been already downloaded to his/her computer. However, you may also configure a link to always start at a certain position. In such a case, our CDN will always generate a truncated video asset that starts at the specified position.

Setting a Start Position for a Flash Video (FLV)

You can programmatically determine the position at which a Flash video (FLV) will be started by setting the ec_seek parameter to the desired position.

Note: If the desired video has been encoded using the H.264 codec, then you should refer to the **Setting the Start Position for H.264 Encoded Videos** section below.

In order to calculate the exact position where a video will start, you will need to use the following formulas:

$$ec_seek = StartPosition * ScrubberBar$$

$$ScrubberBar = File_Size / (Duration / Key_Frame_Interval)$$

A description is provided below for each term in the above formulas.

Term	Description
<i>StartPosition</i>	This term represents the position in the video where the video will start. This position should be specified as the number of seconds from the beginning of the video asset.
<i>ScrubberBar</i>	This term represents the total number of bytes that is represented by 1 second in the scrubber bar. A scrubber bar is the progress bar that allows you to seek within a video. This value can be calculated by the following formula: $File_Size / (Duration / Key_Frame_Interval)$.
<i>File_Size</i>	This term represents the total file size of the video in question. File size should be specified in bytes.
<i>Duration</i>	This term represents the total length of the video in seconds.
<i>Key_Frame_Interval</i>	This term represents the interval at which key frames have been implemented for the desired video. This term should be specified in the number of seconds between key frames.

For example, if you would like to start a video at the 30 second mark, whose file size is 20 MB with a length of 80 seconds and a key frame interval of 2 seconds, then you would perform the following calculations:

$$ScrubberBar = 20971520 / (80 / 2) = 524288$$

$$ec_seek = 30 * 524288 = 15728640$$

The above calculations indicate that in order to start your client's videos at the 30 second mark for that particular video asset, you would need to set `ec_seek` to 15728640 bytes (i.e., `ec_seek=15728640`).

Setting the Start Position for H.264 Encoded Videos

You can programmatically determine the position at which an H.264 encoded video (i.e., MP4, MOV, and F4V) will be started by setting the `ec_seek` parameter. This parameter will need to be set to the number of seconds elapsed from the beginning of the video.

Note: If the desired video has been encoded using the FLV codec, then you should refer to the **Setting a Start Position for a Flash Video (FLV)** section above.

In order to calculate the exact position where a video will start, you will need to use the following formula:

$$ec_seek = Seconds$$

Providing Seek Support for a Custom Flash Player

If you plan on creating a custom Flash player and would like to implement seek support, you will need to make sure that it supports the following:

- The seek parameter must be defined as "ec_seek."
- The ec_seek parameter must be implemented as a query string parameter. Each seek request generated by the Flash player must contain an ec_seek query string parameter.
- The ec_seek parameter must be defined as indicated above for either FLV or H.264 videos.
- If you plan on seeking on videos that have been protected by Token-Based Authentication, you will need to ensure that the ec_seek parameter is appended to the token value. A token value must be the first query string parameter in the requested URL (e.g., http://www.mydomain.com/video.mp4?ak32d0kd0fm12fonmf2&ec_seek=0).

HTTP Live Streaming

Introduction

The HTTP Live Streaming (HLS) solution is designed to allow users to quickly stream a live event and/or on-demand content to iOS devices and any device running QuickTime. It additionally supports the capability to perform adaptive or dynamic streaming. This means that it can broadcast multiple streams of varying bit rate qualities. The media player will then constantly analyze network conditions and CPU usage to choose the stream quality that will provide the best user experience (i.e., high quality video without excessive buffering or stuttering).

HTTP Live Streaming Requirements

This section describes the various requirements needed to stream a live event or on-demand content through our network.

Requirements:

- **Encoder (Live Streaming Only):** Adobe Flash Media Live Encoder 3.2
- **Storage (On-Demand Streaming Only):** CDN storage (i.e., CDN origin server)
- **File Format (On-Demand Streaming Only):** MPEG-4 (MP4)
- **Video Codec:** Video should be encoded using one of the following: H.264 Baseline Level 3.0, Baseline Level 3.1, and Main Level 3.1
- **Audio Codec:** Audio should be encoded using one of the following:
 - HE-AAC or AAC-LC up to 48 kHz, stereo audio
 - MP3 (MPEG-1 Audio Layer 3) 8 kHz to 48 kHz, stereo audio

Note: An additional plug-in may be required for AAC support on the Windows version of Flash Media Live Encoder.

Note: iOS requires the AAC audio codec.

- **Content Delivery Network:** We provide servers that transmit your live event and/or on-demand content to all necessary edge servers for delivery to your end-users. However, you will need to activate the HLS & HDS live and/or on-demand streaming capability from within the MCC. If these options are unavailable, please contact your CDN account manager.

- **Client:** A client will need to meet one of the following requirements to view your stream:
 - iOS 3.0 or later (including iPad and Apple TV)
 - Any computer with Safari 4.0 or later installed
 - Roku 3
 - Limited Android 4.0/4.1+ support

Note: For more detailed service information and additional recommendations, please refer to Apple's documentation on [HTTP Live Streaming](#).


Note: Android 4.0 and 4.1+ provide limited native support for HTTP Live Streaming. [View a sampling of the issues](#) that your viewers may encounter.

Token-Based Authentication

HTTP Live Streaming (Live & On-Demand Streaming) can be secured by Token-Based Authentication under the following circumstances:

- HLS content should not be streamed from a location defined in the **Directories to Authenticate** section of the **Token Auth** page.
- Token-Based Authentication should only be enabled through HTTP Rules Engine. The Token Auth feature determines whether Token-Based Authentication will be enabled on a per request basis.
- Token-Based Authentication should only be enabled on manifest files (i.e., m3u8 and f4m). The easiest way to achieve this is through the URL Path Extension match option.

To create a rule that secures manifest files with Token-Based Authentication

1. From the **Rules Engine** page, set the **Name / Description** option to "Enable Token-Based Authentication on Manifest Files."
2. Select "URL Path Extension" in the option that appears next to the "IF" option.
3. Set the value of the URL Path Extension match option to one of the following:
 - **Secure HLS Only:** m3u8
 - **Secure Both HLS & HDS:** m3u8 f4m
4. Click the  button, which appears next to Features. A new feature option should appear next to it.
5. Select the "Token Auth" option.
6. Set the Token Auth feature to "Enabled."
7. Click **Add** to create this rule.

HTTP Rules Engine

Keep in mind that HTTP Rules Engine can impact this service's functionality. For example, a rule that defines a cache policy on all requests will prevent a player from retrieving an updated manifest file. In turn, this will prevent the player from properly streaming the requested media. Unless you are enabling Token-Based Authentication on a manifest file, it is usually a good idea to modify each rule's match options (e.g., CDN Origin, Customer Origin, Edge CNAME, etc.) to exclude this service.

Keep the following tips in mind when defining or reviewing rules:

- Most rules that apply to all requests will impact this service. Please carefully review all such rules.
- The best way to ensure that a rule will not apply to this service is to match by origin server or by URL.
- If you are matching by origin server or by URL, keep in mind that this service uses the following origin identifiers:
 - **Live Streaming:** 20 and 23 (e.g., /200001 and /230001)
 - **On-Demand Streaming:** 03 (e.g., /030001)

How Does It Work?

This section provides a brief overview on how HTTP Live Streaming works.

Live Streaming

Streaming a live event over HTTP Live Streaming – Live Streaming involves three components, which are an event configuration, an encoder, and a media player. An event configuration allows an encoder to broadcast a stream to our network. In turn, this allows a media player to stream your live event.

Tip: Encoding multiple bit rates for a live event allows a media player to leverage a DVR window through which users can play back earlier content.

On-Demand Streaming

Leveraging the power of HTTP Live Streaming – On-Demand Streaming is as easy as pointing your media player to H.264 video content using a CDN or edge CNAME URL. Our CDN service will take care of packaging and segmenting the video into a format that can be understood by the HTTP Live Streaming-compatible media player.

Setting up HTTP Live Streaming

The configuration of HTTP Live Streaming varies according to whether you would like to stream a live event or on-demand content. Both types of configuration are discussed below.

Note: The Live Streaming & On-Demand Streaming components of HLS must be activated separately. It may take up to an hour before the desired HLS component is activated on your account.

Live Streaming

Setting up HTTP Live Streaming – Live Streaming involves the following steps:

1. Activation of the Live Streaming component of HLS/HDS.
2. Creation of an event configuration.
3. Broadcasting a stream through the use of an encoder.
4. Configuring a media player through which the stream will be played.

HTTP Live Streaming - Live Streaming Activation

A requirement for streaming a live event using HLS is the activation of HLS/HDS – Live Streaming.

To activate HLS – Live Streaming

1. From the **HTTP Large** menu, point to **HTTP Streaming**, and then select **Live HLS & HDS**.
2. Make sure that the **Enable Live HLS & HDS playback** option is marked.


Creating an Event Configuration

In order for an encoder to publish a stream to our network, it needs to know the URL to which the stream should be published. A set of URLs through which an encoder can publish a stream to our ingest servers are automatically generated upon creating an event.

Note: It may take up to an hour before the creation, modification, or deletion of an event takes effect.

Note: Both publishing point and player URLs are case-sensitive. It is recommended that you copy and paste these URLs when configuring your encoder and media player.

The configuration of an event consists of the following items:

Option	Description
Event Name	Provides a description for the type of stream that will be associated with this event configuration.
Instance Name	<p>Defines the name of the application instance. An application instance is typically used to identify your encoder.</p> <hr/> <p>Tip: If you only have a single encoder, then it is recommended to always use the same instance name.</p> <hr/>
Keyframe Interval	<p>Defines the time interval, in seconds, between key frames in the encoded media. A key frame indicates a starting and ending point between different audio/video segments. A media player can transition to different bit rate streams at the end of a segment.</p> <hr/> <p>Important: The key frame interval associated with an event configuration does not affect a stream's frame rate. However, your encoder's key frame frequency must match the value specified for this option. In Adobe FMLE 3.2, the key frame interval is defined by the Keyframe Frequency option.</p> <p>This option can be found by clicking  that appears next to the Format option.</p> <hr/> <p>Tip: Setting this option to a value of 3 or 4 is recommended to achieve optimal encoding and playback performance.</p> <hr/>
Expiration	<p>Defines the expiration date for the event. An event configuration is eligible for deletion at midnight (00:00:00 GMT) on the expiration date. The default value for this option is 30 days from which it was created.</p> <p>This feature is useful for cleaning up old events that are no longer in use. If you plan on reusing the same event, then you should set a longer expiration time.</p>
Encrypt HLS	<p>Determines whether HLS streams associated with this event will undergo AES-128 wire encryption. Encrypted streams can only be decrypted by players that support PHLs (e.g., iOS devices, QuickTime, and Android devices).</p> <hr/> <p>Note: This feature does not require additional CDN or player configuration. A player that supports PHLs can automatically play encrypted streams.</p> <hr/> <p>Note: This option only affects HLS streams. Regardless of this option, HDS streams will be served in an unencrypted format.</p> <hr/>

Option	Description
DVR	<p>Determines whether DVR will be enabled for the event and the length of the window. For the purpose of this document, DVR provides a viewer with the capability to rewind a live stream. The length of time from the present moment that a viewer can rewind a live stream is known as the DVR window. The length of this DVR window can be set from 5 to 180 minutes (i.e., 3 hours).</p> <hr/> <p>Note: A live stream must have a minimum buffer window of 60 seconds. This window exists regardless of whether DVR has been enabled on a stream.</p> <hr/>
Segment Size	<p>Important: This is an advanced setting that requires careful planning. Modifying this setting may cause player incompatibility and playback issues. Before modifying this setting, please consult your media player's documentation.</p> <hr/> <p>Determines the size of the segments that will be generated for the event. Segment size, which is defined in seconds, can be set from 1 - 20 seconds.</p> <p>Default value: 10 seconds</p> <hr/> <p>Note: Apple recommends segments of 10 seconds to achieve a balance between latency, startup time, and network overhead.</p> <hr/>

To create an event configuration


1. View the **Live HLS & HDS** page which can be found under the **HTTP Streaming** sub navigation tab of the **HTTP Large** tab.
2. At the bottom of the page, find the **Event Name** option. Type the name of the event that will be published.
3. Verify that the **Keyframe Interval** option matches the value defined in your encoder.
4. Use the **Encrypt HLS** option to determine whether streams generated for this event should be encrypted.
5. Use the **DVR** option to determine whether DVR will be enabled for this event.
6. Click **Add** to create a new event configuration. Your new event will indicate the available encoder and player URLs.

Reminder: It may take up to an hour before all clients can connect to your newly created event.

Modifying an Event Configuration


Any setting assigned to an event configuration can be modified. However, modifying certain options (i.e., event and instance name) will affect the publishing point and/or player URLs associated with the modified event. These types of changes will require that you update your encoder and/or media players to reflect the new URLs.

To modify an event configuration

1. View the **Live HLS & HDS** page which can be found under the **HTTP Streaming** sub navigation tab of the **HTTP Large** tab.
2. Under the **Event Names** section, click the pencil () that appears to the left of the event configuration that you would like to modify. The settings associated with that event configuration will appear directly below the **Event Names** section.
3. Modify the desired settings.
4. When finished, click **Update** to save your changes. The publishing point and player URLs associated with the modified event configuration will be updated to reflect your changes.

Reminder: It may take up to an hour before the changes to your event configuration take effect.

Deleting an Event Configuration

An event configuration can be deleted by clicking  , which appears to the left of the desired event on the **Live HLS & HDS** page. When prompted, you will need to confirm the deletion of the event configuration.

Warning: An event configuration can be deleted even if there are clients connected to it. The deletion of an event will cause all connections to that stream to be dropped.

Broadcasting Encoded Media

Once you have created an event configuration, you will need to set up your encoder (i.e., Adobe Flash Media Live Encoder 3.2). Basic encoder configuration consists of:

- Defining the location where the encoder can find the desired feed (i.e., audio/video source).
- Defining the audio/video output (e.g., output format, audio/video codec, bit rate levels, etc.).
- Defining the location to which the encoder will output encoded media.

For detailed information on how to configure your encoder, please consult the documentation provided with it.

Defining an Encoder's Video Output

The video output generated by an encoder defines the viewing experience. Two key settings that require special attention are:

- **Format:** HLS requires H.264 format.
- **Bit rate levels:** Define each bit rate level that will be generated by the encoder. When defining bit rate levels, keep in mind that you will need to strike a balance between providing high quality feeds and the amount of bandwidth supported by the computer hosting your encoder. The minimum bandwidth required by an encoder can be calculated by summing up all of the bit rate levels being generated by it.

Defining an Encoder's Output Location

The **Live HLS & HDS** page provides a list of the URLs through which an encoder's media output can be ingested. On this page, each event will display several URLs that correspond to different locations around the world. You should choose the URL corresponding to the location closest to your encoder. This ensures quicker data delivery between your encoder and the CDN ingest point.

Important: In order to prevent unauthorized streams, a Live Authentication key is required to authorize an encoder to publish your stream to our servers. This Live Authentication key must be appended after the stream name (e.g., MyStream?MyLiveAuthenticationKey&adbe-live-event=MyEvent). Since HLS - Live Streaming leverages our Flash Media servers to ingest your encoded media, this Live Authentication key is defined on the **Live Auth** page which can be found on the **Flash** tab.

Important: Encoding multiple bit rates requires that you include the total bit rate value in each stream name. The easiest way to accomplish this is through the use of the %b parameter in the stream name (e.g., MyStream%b?123456&adbe-live-event=event1). For more information, please refer to the following article:

[Adobe Flash Media Live Encoder Parameters and Multi-Bit Rate Encoding.](#)

Note: You can use either a global or a stream key to authenticate your encoder to our ingest servers. If you plan on using stream keys, keep in mind that you must generate a stream key for each bit rate stream (e.g., MyStream750, MyStream500, and MyStream250).

To define an encoder's output location

1. From the **Live HLS & HDS** page, find the desired event and then copy the URL corresponding to the location closest to your encoder.
2. Paste the publishing point URL into the **FMS URL** option.
3. Cut "<streamName>?adbe-live-event=EventName" from the **FMS URL** option.
4. Delete the trailing backslash from the **FMS URL** option. The **FMS URL** option should now look like:

```
rtmp://fso.lax.0001.edgecastcdn.net/200001/default
```

5. Paste the stream name into the **Stream** option.
6. Replace "<streamName>" with the name that you would like to use identify your stream. If you are encoding multiple bit rates, then please append %b to the stream name. For more information, refer to the following article:
[Adobe Flash Media Live Encoder Parameters and Multi-Bit Rate Encoding](#).
7. Insert a Live Authentication key and an ampersand directly after the question mark. The **Stream** option should now look like:

```
MyStream%b?MyLiveAuthenticationKey&adbe-live-event=MyEvent
```

8. Double-check your encoder's media output settings and then start encoding your live stream.

Setting up a Media Player

Point your HLS-compatible media player to the player URL corresponding to the event to which your encoder is publishing a stream.

To construct a player URL

1. From the **Live HLS & HDS** page, copy the **HLS Playback URL** associated with the desired event.
2. Paste it into your media player's source code so that it points to it.
3. Replace "<streamName>" with the name of your stream.
4. If you are encoding multiple bit rates, then you will need to append a comma to the base stream name and then specify a comma-delimited list that identifies each stream.
 - For example, if the encoder's **Stream** option was set to MyStream%b and you are encoding 750, 500, and 250 bit rate streams, then you would append the following value: ,750,500,250,. The resulting stream name would look like: MyStream,750,500,250,.m3u8.

A sample CDN URL is provided below.

```
http://wpc.0001.edgecastcdn.net/hls-  
live/200001/MyInstance/MyEvent/MyStream,750,500,250,.m3u8
```

Note: If a stream name contain a suffix, then you may append it after the last comma (e.g., MyStream,750,500,250,kbps.m3u8).

Note: For additional information on how to set the player URL, please refer to the **Single vs. Multiple Streams** section below.

To generate an edge CNAME version of the player URL

1. Create an edge CNAME configuration.
 - i. Point the new edge CNAME configuration to an HLS or HDS Live Streaming origin.
 - ii. Add the corresponding CNAME record via a DNS service provider.
2. Manually construct the URL.
 - i. Copy the edge CNAME URL created above.
 - ii. Append the remnant of the HLS player URL to it.

A sample edge CNAME URL (HLS Live Streaming) is provided below.

```
http://hls.mydomain.com/MyInstance/MyEvent/MyStream,750,500,250,.m3u8
```

Single vs. Multiple Streams

A media player can request a stream generated from a single H.264 bit rate stream or it can analyze a user's environment at frequent intervals and dynamically choose from a set of streams of varying quality the one that will provide the best user experience. The number of streams referenced in the player URL determines the media player's behavior.

Single Stream

Stream a single H.264 asset by modifying the player URL to reflect the stream name defined in the encoder's **Stream** option.

Multiple Streams

If multiple streams are referenced in the player URL, a media player will dynamically analyze a user's bandwidth and CPU usage and then stream the bit rate quality that will provide the optimal user experience. This capability requires the following:

- An encoder must publish several streams of varying quality.
- The base stream name defined in the encoder must be the same for all streams. Use the %b parameter to identify each stream by its total bit rate level.
- Each bit rate level must be indicated in the player URL. A media player's initial request will be for the first bit rate specified in the player URL.

Player URL Format

The player URL for multiple streams is similar to the one used for a single stream. The main difference between the two URLs is that the one for multiple streams must identify each stream generated by your encoder. Multiple streams can be specified within a single URL by following specific file naming conventions.

Keep the following items in mind when identifying different streams in a player URL.

- The terms *prefix* and *suffix* refer to the portions of the stream name that appear before and after, respectively, the parameter.
- All streams must have a common prefix (e.g., **video100**, **video200**, and **video300**).
- The end of the prefix should be indicated with a comma. This should be followed by a comma-delimited list of the bit rate values used to identify each unique stream (e.g., `video,100,200,300,.m3u8`).
- A suffix identifies the portion of the stream name that extends beyond the comma-delimited list of bit rate values used to identify each unique stream (e.g., `video200kbps`). If there is a suffix, then you may append it after the last comma (e.g., `video,100,200,300,kbps.m3u8`).

The following CDN URL illustrates the proper syntax for streams that contain a suffix:

```
http://wpc.xxxx.edgecastcdn.net/hls-  
live/20xxxx/instance/event/prefix,bitrate1,bitrate2,bitrateN,suffix.m3u8
```

The following CDN URL illustrates the proper syntax for streams that do not contain a suffix:

```
http://wpc.xxxx.edgecastcdn.net/hls-  
live/20xxxx/instance/event/prefix,bitrate1,bitrate2,bitrateN,.m3u8
```

Archiving a Live Event

A live event can be recorded as it is being streamed and then archived to CDN storage. This process is known as Server-Side Archiving. One use of this feature is to provide on-demand capabilities for content that was originally streamed as a live event.

Tip: We offer various streaming technologies (i.e., Flash Media Streaming, HLS, HDS, and HTTP Progressive Download) through which archived content can be played back.

Encoder Configuration

The Server-Side Archiving feature does not require a different naming convention for your encoder's **Stream** option. In fact, it is important that it not deviate from the following pattern:

```
StreamName?LiveAuthenticationKey&adbe-live-event=EventName
```

Note: Adding an mp: prefix or a file name extension to the stream name will impact media playback.

Server-Side Archiving Activation

The Server-Side Archiving feature must be activated before you can archive your live streams. This feature can be activated by marking the **Enable Server Side Archiving** option from the **Server Side Archiving** page, which can be found by clicking the **Flash** tab, the **Advanced Settings** sub navigation tab, and then selecting it from the side navigation menu.

Note: The **Enable Server Side Archiving** option toggles the activation of the Server-Side Archiving feature for the following solutions: Flash Media Streaming, HLS, and HDS.

Storage Location

Live streams are archived on a CDN origin server in a folder named after the live event's instance name. This folder can be found in the root folder of your CDN storage account.

For example, if you set your encoder to output encoded media to:

```
rtmp://fso.lax.xxx.edgecastcdn.net/20xxx/default
```

Then the live stream would be archived to the following relative path on the CDN origin server:

```
/default
```

Reminder: You can access content stored on a CDN origin server using a third-party FTP client or the embedded FTP client provided on the **File Manager** page, which can be found on the **Storage** tab.

File Format

A live event is always archived as an MP4 file.

Appending or Overwriting Archived Content

When archiving your streaming media, you have the option to either append or overwrite your previously recorded files when your stream is interrupted. This capability is controlled by the **Enable Appending** option from the [Advanced Settings – Server Side Archiving](#) page of the **Flash** tab in the MCC. By default, the name for your archived file will be:

- *StreamName.mp4*

If the specified ingest server is unable to resume encoding after a stream interruption, then a naming convention slightly different from the one specified above will be used. This provides a safeguard against data loss due to duplicate filenames. The naming convention for files archived under this circumstance is provided below.

- *StreamName.#.mp4* (e.g., Demo.1.mp4)

On-Demand Streaming

Setting up HTTP Live Streaming – On-Demand Streaming involves the following steps:

1. Activation of the On-Demand Streaming component of HLS/HDS.
2. Uploading the desired H.264 video to CDN storage.
3. Configuring a media player through which the stream will be played.

HTTP Live Streaming – On-Demand Streaming Activation

A requirement for streaming on-demand content using HLS is the activation of HLS – On-Demand Streaming.

Note: For information on how to encrypt on-demand content, please refer to the **Stream Encryption** section below.

To activate HLS – On-Demand Streaming

1. From the **HTTP Large** menu, point to **HTTP Streaming**, and then select **On-Demand HLS & HDS**.
2. Make sure that the **Enable On-Demand HLS & HDS playback** option is marked.

Uploading Video Content

HLS – On-Demand Streaming is only compatible with H.264 videos stored on CDN storage.

Setting Up a Media Player

Point your HLS-compatible media player to the appropriate player URL.

To construct a player URL

1. Copy a base CDN or edge CNAME URL that points to HTTP Live Streaming – On-Demand Streaming. The specified player URL should use "03" for the origin identifier instead of "00."
 - **CDN URL:** [On-Demand HLS & HDS](#) page (**HTTP Streaming** sub navigation tab)
 - **Edge CNAME URL:** Before copying an edge CNAME URL from the [HTTP Large Object](#) page, make sure that it has been configured to point to "On-Demand HLS Origin."
2. Perform one of the following:
 - **Single Stream:** Append ".m3u8" to the end of the URL. A sample CDN URL is provided below.

```
http://wpc.0001.edgecastcdn.net/030001/videos/fly.mp4.m3u8
```

- **Multiple Streams:** Append a comma to the base filename and then specify a comma-delimited list of bit rate levels. A comma should also be appended after the last specified bit rate. After which, you should append ".m3u8" to the end of the URL. A sample CDN URL is provided below.

```
http://wpc.0001.edgecastcdn.net/030001/videos/fly,110,400,650,.mp4.m3u8
```

Note: If the filename for the desired set of videos contain a suffix, then you may append it after the last comma (e.g., fly,110,400,650,Kbps.mp4.m3u8).

Note: For additional information on how to set the player URL, please refer to the **Single vs. Multiple Streams** section below.

Note: For information on how to upload content to CDN storage, please refer to the **CDN Origin Servers** section in the **Configuring an Origin Server** chapter of the **HTTP Large Administration Guide**.

Single vs. Multiple Streams

A media player can request a stream generated from a single H.264 asset or it can analyze a user's environment at frequent intervals and dynamically choose from a set of streams of varying quality the one that will provide the best user experience. The number of H.264 assets referenced in the player URL determines the media player's behavior.

Single Stream

Stream a single H.264 asset by modifying the CDN URL that points to it as indicated below.

1. Change the origin identifier from "00" to "03."
2. Append "m3u8" after the file name extension.

Point the desired media player to the updated CDN URL. The proper syntax for the player URL is defined below.

Player URL Format:

```
http://wpc.xxxx.edgecastcdn.net/03xxxx/path/filename.ext.m3u8
```

Sample CDN URL (HTTP Large):

```
http://wpc.0001.edgecastcdn.net/000001/videos/fly_100Kbps.mp4
```

Sample Player URL:

```
http://wpc.0001.edgecastcdn.net/030001/videos/fly_100Kbps.mp4.m3u8
```

Multiple Streams

If multiple H.264 assets are referenced in the player URL, a media player will dynamically analyze a user's bandwidth and CPU usage and then stream the bit rate quality that will provide the optimal user experience. This capability requires the following:

- An H.264 asset for each desired bit rate stream should be stored in CDN storage. The location in CDN storage doesn't matter as long as they are all stored in the same directory.
- The filename for each H.264 asset should indicate its corresponding bit rate quality in Kbps (e.g., 100, 200, or 400). The bit rate quality is the only difference allowed in the file naming convention.
- Each bit rate stream should be indicated in the player URL. A media player's initial request will be for the first bit rate specified in the player URL.

Player URL Format

The player URL for multiple streams is similar to the one used for a single stream. The main difference between the two URLs is that the one for multiple streams must identify each asset from which a stream can be generated. Multiple assets can be specified within a single URL by following specific file naming conventions.

Keep the following items in mind when identifying content of varying bit rate levels in a player URL.

- The terms *prefix* and *suffix* refer to the portions of the filename that appear before and after, respectively, the bit rate level.
- All filenames must have a common prefix (e.g., **video**100.mp4, **video**200.mp4, and **video**300.mp4).
- The end of the prefix should be indicated with a comma. This should be followed by the bit rate level for each desired stream as a comma-delimited list in the filename (e.g., video,100,200,300.mp4).
- Each specified bit rate stream should only include the bit rate quality in Kbps. Including any other data or using different units will generate a malformed playlist.
- A suffix identifies the portion of the filename that extends beyond the bit rate level (e.g., video100**kbps**.mp4, video200**kbps**.mp4, and video300**kbps**.mp4). If there is a suffix, then you may append it after the last comma (e.g., video,100,200,300,kbps.mp4).

The following CDN URL illustrates the proper syntax for filenames that contain a suffix:

```
http://wpc.xxxx.edgecastcdn.net/03xxxx/path/prefix,bitrate1,bitrate2,bitrateN,suffix.ext.m3u8
```

The following CDN URL illustrates the proper syntax for filenames that do not contain a suffix:

```
http://wpc.xxxx.edgecastcdn.net/03xxxx/path/prefix,bitrate1,bitrate2,bitrateN,.ext.m3u8
```

Example

This sample scenario illustrates how to construct a player URL that can reference multiple H.264 assets.

H.264 Assets (CDN URLs)

```
http://wpc.0001.edgecastcdn.net/000001/videos/fly_100Kbps.mp4
```

```
http://wpc.0001.edgecastcdn.net/000001/videos/fly_200Kbps.mp4
```

```
http://wpc.0001.edgecastcdn.net/000001/videos/fly_300Kbps.mp4
```

Sample Player URL:

```
http://wpc.0001.edgecastcdn.net/030001/videos/fly_,100,200,300,Kbps.mp4.m3u8
```

Stream Encryption

AES-128 wire encryption can be applied to HLS streams generated for your live events and on-demand content. Encrypted streams can only be decrypted by players that support PHLS (e.g., iOS devices, QuickTime, and Android devices). Players that do not support PHLS will be unable to play back encrypted streams.

Note: The playback of an encrypted stream can be performed by a PHLS-compatible player without the need for additional player configuration.

Note: Only HLS streams can be encrypted at this time. This means that it may be possible to download or stream your content using a different streaming technology (e.g., HDS or Flash Media Streaming). If you have sensitive streaming content that requires additional protection, then it is highly recommended that you leverage HTTP Rules Engine to deny all non-HLS requests for H.264 assets. For more information, please contact your CDN account manager.

PHLS Configuration (Live Streaming)

An event's configuration determines whether its streams will be encrypted. Specifically, the **Encrypt HLS** option toggles whether AES-128 wire encryption will be applied to all streams associated with the event.

PHLS Configuration (On-Demand Streaming)

The **Protected Directories for PHLS** section allows you to define one or more locations that will generate encrypted streams from your on-demand content. A location can be defined by specifying the relative path to the desired folder. The starting point for this relative path is defined below:

URL Type	Relative Path (Starting Point)
CDN URL (CDN Origin)	<p>Specify a relative path that starts directly after the content access point (i.e., /030001).</p> <p>Sample URL:</p> <p><code>http://wpc.0001.edgecastcn.net/030001/mybusiness/videos/fly.mp4</code></p> <p>In the above sample URL, the gray text indicates what should be excluded when securing a location. This sample request can be secured by any of the following configurations:</p> <ul style="list-style-type: none">• /• /mybusiness• /mybusiness/videos
Edge CNAME URL (CDN Origin)	<p>Specify a relative path that starts directly after the hostname.</p> <p>Sample URL:</p> <p><code>http://www.domain.com/mybusiness/videos/fly.mp4</code></p> <p>In the above sample URL, the gray text indicates what should be excluded when securing a location. This sample request can be secured by any of the following configurations:</p> <ul style="list-style-type: none">• /• /mybusiness• /mybusiness/videos

Note: The path to a protected folder always starts with a forward slash (/).

Note: It may take up to an hour before a new location is fully protected.

Note: Wildcard characters (e.g., *) are not supported when setting up protected directories.

Scope

When choosing which folders will generate encrypted streams, keep in mind that this feature is applied recursively to that folder. This means that streaming on-demand content residing in the specified folder or its subfolders through HLS will generate encrypted streams.

To encrypt all on-demand content streamed through HLS

1. Navigate to the **On-Demand HLS & HDS** page.
2. Make sure that the **Enable On-Demand PHLS** option is enabled.
3. Set the **New** option, which can be found in the **Protected Directories for PHLS** section, to forward slash (/).
4. Click **Add**.

Purging Live & On-Demand Content

Live and on-demand content can be purged from our network. A purge request will remove the cached version of the live or on-demand content from our edge servers.

Below are a couple of scenarios under which you may wish to purge content:

- **Live Event:** A purge request can be a quick and easy way to prevent users from viewing cached portions of your live stream after your live event has completed.
- **On-Demand Content:** Purge this type of content when you would like to distribute updated content without modifying your existing links. Simply update the source media on the origin server and then purge the cached content. This will force our edge servers to forward requests for that content to your origin server.

This streaming solution generates the following assets for each requested live or on-demand stream:

- General and bit rate-specific manifest files
- Bit-rate specific fragments

This means that purging the player URL will not purge the corresponding live or on-demand content. In order to properly purge this content, you will need to perform one of the following:

- Purge the parent folder:

Live Event:

```
http://wpc.xxxx.edgecastcdn.net/23xxxx/InstanceName/EventName/*
```

On-Demand Content:

```
http://wpc.xxxx.edgecastcdn.net/03xxxx/path/*
```

- Purge all variations of the base file name:

On-Demand Content:

```
http://wpc.xxxx.edgecastcdn.net/03xxxx/path/basefilename*
```

Keep the following information in mind when purging content:

- The recommended approach for purging varies by stream method:
 - **Live Event:** Purge the parent folder.
 - **On-Demand Content:** Purge all variations of the base file name. This will ensure that other content stored in the parent folder will not be inadvertently purged.
- It is important to specify a CDN or edge CNAME URL that points to HLS (i.e., 23 or 03). Otherwise, the referenced content will not be purged.
- If you plan on purging a live event, keep in mind that the CDN URL used to play back the stream will not match the purge URL. A sample CDN and purge URL is provided below.

Sample Player URL (CDN URL):

```
http://wpc.0001.edgecastcdn.net/hls-live/200001/default/event1/mystream.m3u8
```

Sample Purge URL:

```
http://wpc.0001.edgecastcdn.net/230001/default/event1/*
```

- The purge instructions defined in this topic apply regardless of whether the player URL references a single or multiple H.264 assets.

Purging a Live Event Example

Provided below is an example of how to purge a live event.

Sample Player URL:

```
http://wpc.0001.edgecastcdn.net/hls-live/200001/default/myevent1/fly,110,400,650,.m3u8
```

Purge the parent folder:

```
http://wpc.0001.edgecastcdn.net/230001/default/myevent1/*
```

Purging On-Demand Content Example

Provided below are examples of two different methods by which on-demand content can be purged.

Sample Player URL:

```
http://wpc.0001.edgecastcdn.net/030001/videos/fly,110,400,650,.mp4.m3u8
```

Purge the parent folder:

```
http://wpc.0001.edgecastcdn.net/030001/videos/*
```

Purge all variations of the base file name:

```
http://wpc.0001.edgecastcdn.net/030001/videos/fly*
```

HTTP Dynamic Streaming

Introduction

The HTTP Dynamic Streaming (HDS) solution is designed to allow users to quickly stream a live event and/or on-demand content using Adobe products. It additionally supports the capability to perform adaptive or dynamic streaming. This means that it can broadcast multiple streams of varying bit rate qualities. The media player will then constantly analyze network conditions and CPU usage to choose the stream quality that will provide the best user experience (i.e., high quality video without excessive buffering or stuttering).

HTTP Dynamic Streaming Requirements

This section describes the various requirements needed to stream on-demand content through our network.

Requirements:

- **Encoder (Live Streaming Only):** Adobe Flash Media Live Encoder 3.2
- **Storage (On-Demand Streaming Only):** CDN storage (i.e., CDN origin server)
- **File Format (On-Demand Streaming Only):** MPEG-4 (MP4)
- **Video Codec:** H.264
- **Audio Codec:** Audio should be encoded using one of the following:
 - HE-AAC or AAC-LC up to 48 kHz, stereo audio
 - MP3 (MPEG-1 Audio Layer 3) 8 kHz to 48 kHz, stereo audio
- **Content Delivery Network:** We provide servers that transmit your live event and/or on-demand content to all necessary edge servers for delivery to your end-users. However, you will need to activate the HLS & HDS live and/or on-demand streaming capability from within the MCC. If these options are unavailable, please contact your CDN account manager.
- **Client:** A client will need to meet one of the following requirements to view your stream:
 - Adobe Flash Player 10.1 and above
 - Adobe AIR 2 and above


Note: For more detailed information and additional recommendations, please refer to Adobe's documentation on [HTTP Dynamic Streaming](#).

Token-Based Authentication

HTTP Dynamic Streaming (Live & On-Demand Streaming) can be secured by Token-Based Authentication under the following circumstances:

- HDS content should not be streamed from a location defined in the **Directories to Authenticate** section of the **Token Auth** page.
- Token-Based Authentication should only be enabled through HTTP Rules Engine. The Token Auth feature determines whether Token-Based Authentication will be enabled on a per request basis.
- Token-Based Authentication should only be enabled on manifest files (i.e., m3u8 and f4m). The easiest way to achieve this is through the URL Path Extension match option.

To create a rule that secures manifest files with Token-Based Authentication

1. From the **Rules Engine** page, set the **Name / Description** option to "Enable Token-Based Authentication on Manifest Files."
2. Select "URL Path Extension" in the option that appears next to the "IF" option.
3. Set the value of the URL Path Extension match option to one of the following:
 - **Secure HDS Only:** f4m
 - **Secure Both HLS & HDS:** m3u8 f4m
4. Click the  button, which appears next to Features. A new feature option should appear next to it.
5. Select the "Token Auth" option.
6. Set the Token Auth feature to "Enabled."
7. Click **Add** to create this rule.

HTTP Rules Engine

Keep in mind that HTTP Rules Engine can impact this service's functionality. For example, a rule that defines a cache policy on all requests will prevent a player from retrieving an updated manifest file. In turn, this will prevent the player from properly streaming the requested media. Unless you are enabling Token-Based Authentication on a manifest file, it is usually a good idea to modify each rule's match options (e.g., CDN Origin, Customer Origin, Edge CNAME, etc.) to exclude this service.

Keep the following tips in mind when defining or reviewing rules:

- Most rules that apply to all requests will impact this service. Please carefully review all such rules.
- The best way to ensure that a rule will not apply to this service is to match by origin server or by URL.
- If you are matching by origin server or by URL, keep in mind that this service uses the following origin identifiers:
 - **Live Streaming:** 20 and 22 (e.g., /200001)
 - **On-Demand Streaming:** 02 (e.g., /020001)

How Does It Work?

This section provides a brief overview on how HTTP Dynamic Streaming works.

Live Streaming

Streaming a live event over HTTP Dynamic Streaming – Live Streaming involves three components, which are an event configuration, an encoder, and a media player. An event configuration allows an encoder to broadcast a stream to our network. In turn, this allows a media player to stream your live event.

Tip: Encoding multiple bit rates for a live event allows a media player to leverage a DVR window through which users can play back earlier content.

On-Demand Streaming

Leveraging the power of HTTP Dynamic Streaming is as easy as pointing your media player to H.264 video content using a CDN URL. Our CDN service will take care of packaging and segmenting the video into a format that can be understood by the HTTP Dynamic Streaming-compatible media player.

Setting up HTTP Dynamic Streaming

The configuration of HTTP Dynamic Streaming varies according to whether you would like to stream a live event or on-demand content. Both types of configuration are discussed below.

Note: The Live Streaming & On-Demand Streaming components of HDS must be activated separately. It may take up to an hour before the desired HDS component is activated on your account.

Live Streaming

Setting up HTTP Dynamic Streaming – Live Streaming involves the following steps:

1. Activation of the Live Streaming component of HLS/HDS.
2. Creation of an event configuration.
3. Broadcasting a stream through the use of an encoder.
4. Configuring a media player through which the stream will be played.

HTTP Dynamic Streaming - Live Streaming Activation

A requirement for streaming a live event using HDS is the activation of HLS/HDS – Live Streaming.

To activate HDS – Live Streaming

1. From the **HTTP Large** menu, point to **HTTP Streaming**, and then select **Live HLS & HDS**.
2. Make sure that the **Enable Live HLS & HDS playback** option is marked.


Creating an Event Configuration

In order for an encoder to publish a stream to our network, it needs to know the URL to which the stream should be published. A set of URLs through which an encoder can publish a stream to our ingest servers are automatically generated upon creating an event.

Note: It may take up to an hour before the creation, modification, or deletion of an event takes effect.

Note: Both publishing point and player URLs are case-sensitive. It is recommended that you copy and paste these URLs when configuring your encoder and your media player.

The configuration of an event consists of the following items:

Option	Description
Event Name	Provides a description for the type of stream that will be associated with this event configuration.
Instance Name	<p>Defines the name of the application instance. An application instance is typically used to identify your encoder.</p> <hr/> <p>Tip: If you only have a single encoder, then it is recommended to always use the same instance name.</p> <hr/>
Keyframe Interval	<p>Defines the time interval, in seconds, between key frames in the encoded media. A key frame indicates a starting and ending point between different segments. A media player can transition to different bit rate streams at the end of a segment.</p> <hr/> <p>Important: The key frame interval associated with an event configuration does not affect a stream's frame rate. However, your encoder's key frame frequency must match the value specified for this option. In Adobe FMLE 3.2, the key frame interval is defined by the Keyframe Frequency option.</p> <p>This option can be found by clicking  that appears next to the Format option.</p> <hr/> <p>Tip: Setting this option to a value of 3 or 4 is recommended to achieve optimal encoding and playback performance.</p> <hr/>
Expiration	<p>Defines the expiration date for the event. An event configuration is eligible for deletion at midnight (00:00:00 GMT) on the expiration date. The default value for this option is 30 days from which it was created.</p> <p>This feature is useful for cleaning up old events that are no longer in use. If you plan on reusing the same event, then you should set a longer expiration time.</p>
DVR	<p>Determines whether DVR will be enabled for the event and the length of the window. For the purpose of this document, DVR provides a viewer with the capability to rewind a live stream. The length of time from the present moment that a viewer can rewind a live stream is known as the DVR window. The length of this DVR window can be set from 5 to 165 minutes (i.e., 2 hours and 45 minutes).</p> <hr/> <p>Note: A live stream must have a minimum buffer window of 60 seconds. This window exists regardless of whether DVR has been enabled on a stream.</p> <hr/>

Option	Description
Segment Size	<p data-bbox="548 254 1425 394">Important: This is an advanced setting that requires careful planning. Modifying this setting may cause player incompatibility and playback issues. Before modifying this setting, please consult your media player's documentation.</p> <hr/> <p data-bbox="548 436 1425 499">Determines the size of the segments that will be generated for the event. Segment size, which is defined in seconds, can be set from 1 - 20 seconds.</p> <p data-bbox="548 520 862 552">Default value: 10 seconds</p> <hr/> <p data-bbox="548 583 1398 646">Note: Apple recommends segments of 10 seconds to achieve a balance between latency, startup time, and network overhead.</p> <hr/>

To create an event configuration


1. View the **Live HLS & HDS** page which can be found under the **HTTP Streaming** sub navigation tab of the **HTTP Large** tab.
2. At the bottom of the page, find the **Event Name** option. Type the name of the event that will be published.
3. Verify that the **Keyframe Interval** option matches the value defined in your encoder.
4. Use the **DVR** option to determine whether DVR will be enabled for this event.
5. Click **Add** to create a new event configuration. Your new event will indicate the available encoder and player URLs.

Reminder: It may take up to an hour before all clients can connect to your newly created event.

Modifying an Event Configuration


Any setting assigned to an event configuration can be modified. However, modifying certain options (i.e., event and instance name) will affect the publishing point and/or player URLs associated with the modified event. These types of changes will require that you update your encoder and/or media players to reflect the new URLs.

To modify an event configuration

1. View the **Live HLS & HDS** page which can be found under the **HTTP Streaming** sub navigation tab of the **HTTP Large** tab.
2. Under the **Event Names** section, click the pencil () that appears to the left of the event configuration that you would like to modify. The settings associated with that event configuration will appear directly below the **Event Names** section.
3. Modify the desired settings.
4. When finished, click **Update** to save your changes. The publishing point and player URLs associated with the modified event configuration will be updated to reflect your changes.

Reminder: It may take up to an hour before the changes to your event configuration take effect.

Deleting an Event Configuration

An event configuration can be deleted by clicking  , which appears to the left of the desired event on the **Live HLS & HDS** page. When prompted, you will need to confirm the deletion of the event configuration.

Warning: An event can be deleted even if there are clients connected to it. The deletion of an event will cause all connections to that stream to be dropped.

Broadcasting Encoded Media

Once you have created an event configuration, you will need to set up your encoder (i.e., Adobe Flash Media Live Encoder 3.2). Basic encoder configuration consists of:

- Defining the location where the encoder can find the desired feed (i.e., audio/video source).
- Defining the audio/video output (e.g., output format, audio/video codec, bit rate levels, etc.).
- Defining the location to which the encoder will output encoded media.

For detailed information on how to configure your encoder, please consult the documentation provided with it.

Defining an Encoder's Video Output

The video output generated by an encoder defines the viewing experience. Two key settings that require special attention are:

- **Format:** HDS requires H.264 format.
- **Bit rate levels:** Define each bit rate level that will be generated by the encoder. When defining bit rate levels, keep in mind that you will need to strike a balance between providing high quality feeds and the amount of bandwidth supported by the computer hosting your encoder. The minimum bandwidth required by an encoder can be calculated by summing up all of the bit rate levels being generated by it.

Defining an Encoder's Output Location

The **Live HLS & HDS** page provides a list of the URLs through which an encoder's media output can be ingested. On this page, each event will display several URLs that correspond to different locations around the world. You should choose the URL corresponding to the location closest to your encoder. This ensures quicker data delivery between your encoder and the CDN ingest point.

Important: In order to prevent unauthorized streams, a Live Authentication key is required to authorize an encoder to publish your stream to our servers. This Live Authentication key must be appended after the stream name (e.g., MyStream?MyLiveAuthenticationKey&adbe-live-event=MyEvent). Since HDS - Live Streaming leverages our Flash Media servers to ingest your encoded media, this Live Authentication key is defined on the **Live Auth** page which can be found on the **Flash** tab.

Important: Encoding multiple bit rates requires that you include the total bit rate value in each stream name. The easiest way to accomplish this is through the use of the %b parameter in the stream name (e.g., MyStream%b?123456&adbe-live-event=event1). For more information, please refer to the following article:

[Adobe Flash Media Live Encoder Parameters and Multi-Bit Rate Encoding.](#)

Note: You can use either a global or a stream key to authenticate your encoder to our ingest servers. If you plan on using stream keys, keep in mind that you must generate a stream key for each bit rate stream (e.g., MyStream750, MyStream500, and MyStream250).

To define an encoder's output location

1. From the **Live HLS & HDS** page, find the desired event and then copy the URL corresponding to the location closest to your encoder.
2. Paste the publishing point URL into the **FMS URL** option.
3. Cut "<streamName>?adbe-live-event=*EventName*" from the **FMS URL** option.
4. Delete the trailing backslash from the **FMS URL** option. The **FMS URL** option should now look like:

```
rtmp://fso.lax.0001.edgecastcdn.net/200001/default
```

5. Paste the stream name into the **Stream** option.
6. Replace "<streamName>" with the name that you would like to use identify your stream. If you are encoding multiple bit rates, then please append %b to the stream name. For more information, refer to the following article:
[Adobe Flash Media Live Encoder Parameters and Multi-Bit Rate Encoding](#).
7. Insert a Live Authentication key and an ampersand directly after the question mark. The **Stream** option should now look like:

```
MyStream%b?MyLiveAuthenticationKey&adbe-live-event=MyEvent
```

8. Double-check your encoder's media output settings and then start encoding your live stream.

Setting up a Media Player

Point your HDS-compatible media player to the player URL corresponding to the event to which your encoder is publishing a stream.

To construct a player URL

1. From the **Live HLS & HDS** page, copy the **HLS Playback URL** associated with the desired event.
2. Paste it into your media player's source code so that it points to it.
3. Replace "<streamName>" with the name of your stream.
4. If you are encoding multiple bit rates, then you will need to append a comma to the base stream name and then specify a comma-delimited list that identifies each stream.
 - For example, if the encoder's **Stream** option was set to MyStream%b and you are encoding 750, 500, and 250 bit rate streams, then you would append the following value: ,750,500,250,. The resulting stream name would look like: MyStream,750,500,250,.f4m.

A sample CDN URL is provided below.

```
http://wpc.0001.edgecastcdn.net/hds-  
live/200001/MyInstance/MyEvent/MyStream,750,500,250,.f4m
```

Note: If the stream name contain a suffix, then you may append it after the last comma (e.g., MyStream,750,500,250,kbps.f4m).

Note: For additional information on how to set the player URL, please refer to the **Single vs. Multiple Streams** section below.

Single vs. Multiple Streams

A media player can request a stream generated from a single H.264 bit rate stream or it can analyze a user's environment at frequent intervals and dynamically choose from a set of streams of varying quality the one that will provide the best user experience. The number of streams referenced in the player URL determines the media player's behavior.

Single Stream

Stream a single H.264 asset by modifying the player URL to reflect the stream name defined in the encoder's **Stream** option.

Multiple Streams

If multiple streams are referenced in the player URL, a media player will dynamically analyze a user's bandwidth and CPU usage and then stream the bit rate quality that will provide the optimal user experience. This capability requires the following:

- An encoder must publish several streams of varying quality.
- The base stream name defined in the encoder must be the same for all streams. Use the %b parameter to identify each stream by its bit rate level.
- Each bit rate level must be indicated in the player URL. A media player's initial request will be for the first bit rate specified in the player URL.

Player URL Format

The player URL for multiple streams is similar to the one used for a single stream. The main difference between the two URLs is that the one for multiple streams must identify each stream generated by your encoder. Multiple streams can be specified within a single URL by following specific file naming conventions.

Keep the following items in mind when identifying different streams in a player URL.

- The terms *prefix* and *suffix* refer to the portions of the stream name that appear before and after, respectively, the parameter.
- All streams must have a common prefix (e.g., **video100**, **video200**, and **video300**).
- The end of the prefix should be indicated with a comma. This should be followed by a comma-delimited list of bit rate values used to identify each unique stream (e.g., video,100,200,300,.f4m).
- A suffix identifies the portion of the stream name that extends beyond the comma-delimited list of bit rate values used to identify each unique stream (e.g., MyStream200**kbps**). If there is a suffix, then you may append it after the last comma (e.g., video,100,200,300,**kbps**.f4m).

The following CDN URL illustrates the proper syntax for filenames that contain a suffix:

```
http://wpc.xxxx.edgecastcdn.net/hds-  
live/20xxxx/instance/event/prefix,value1,value2,valueN,suffix.f4m
```

The following CDN URL illustrates the proper syntax for filenames that do not contain a suffix:

```
http://wpc.xxxx.edgecastcdn.net/hds-  
live/20xxxx/instance/event/prefix,value1,value2,valueN,.f4m
```

Archiving a Live Event

A live event can be recorded as it is being streamed and then archived to CDN storage. This process is known as Server-Side Archiving. One use of this feature is to provide on-demand capabilities for content that was originally streamed as a live event.

Tip: We offer various streaming technologies (i.e., Flash Media Streaming, HLS, HDS, and HTTP Progressive Download) through which archived content can be played back.

Encoder Configuration

The Server-Side Archiving feature does not require a different naming convention for your encoder's **Stream** option. In fact, it is important that it not deviate from the following pattern:

```
StreamName?LiveAuthenticationKey&adbe-live-event=EventName
```

Note: Adding a mp: prefix or a file name extension to the stream name will impact media playback.

Server-Side Archiving Activation

The Server-Side Archiving feature must be activated before you can archive your live streams. This feature can be activated by marking the **Enable Server Side Archiving** option from the **Server Side Archiving** page, which can be found by clicking the **Flash** tab, the **Advanced Settings** sub navigation tab, and then selecting it from the side navigation menu.

Storage Location

Live streams are archived on a CDN origin server in a folder named after the live event's instance name. This folder can be found in the root folder of your CDN storage account.

For example, if you set your encoder to output encoded media to:

```
rtmp://fso.lax.xxxx.edgecastcdn.net/20xxxx/default
```

Then the live stream would be archived to the following relative path on the CDN origin server:

```
/default
```

Reminder: You can access content stored on a CDN origin server using a third-party FTP client or the embedded FTP client provided on the **File Manager** page, which can be found on the **Storage** tab.

File Format

A live event is always archived as an MP4 file.

Appending or Overwriting Archived Content

When archiving your streaming media, you have the option to either append or overwrite your previously recorded files when your stream is interrupted. This capability is controlled by the

Enable Appending option from the [Advanced Settings – Server Side Archiving](#) page of the **Flash** tab in the MCC. By default, the name for your archived file will be:

- *StreamName.mp4*

If the specified ingest server is unable to resume encoding after a stream interruption, then a naming convention slightly different from the one specified above will be used. This provides a safeguard against data loss due to duplicate filenames. The naming convention for files archived under this circumstance is provided below.

- *StreamName.#.mp4* (e.g., Demo.1.mp4)

On-Demand Streaming

Setting up HTTP Dynamic Streaming – On-Demand Streaming involves the following steps:

1. Activation of the On-Demand Streaming component of HLS/HDS.
2. Uploading the desired H.264 video to CDN storage.
3. Configuring a media player through which the stream will be played.

HTTP Dynamic Streaming – On-Demand Streaming Activation

A requirement for streaming on-demand content using HDS is the activation of HDS – On-Demand Streaming.

To activate HDS – On-Demand Streaming

1. From the **HTTP Large** menu, point to **HTTP Streaming**, and then select **On-Demand HLS & HDS**.
2. Make sure that the **Enable On-Demand HLS & HDS playback** option is marked.

Uploading Video Content

HDS – On-Demand Streaming is only compatible with H.264 videos stored on CDN storage.

Setting Up a Media Player

Point your HDS-compatible media player to the appropriate player URL.

To construct a player URL

1. Copy a base CDN URL that points to HTTP Dynamic Streaming – On-Demand Streaming. The specified player URL should use "02" for the origin identifier instead of "00."

- **CDN URL:** [On-Demand HLS & HDS](#) page (**HTTP Streaming** sub navigation tab)

2. Perform one of the following:

- **Single Stream:** Append ".f4m" to the end of the URL. A sample CDN URL is provided below.

```
http://wpc.0001.edgecastcdn.net/020001/videos/fly.mp4.f4m
```

- **Multiple Streams:** Append a comma to the base filename and then specify a comma-delimited list of bit rate levels. A comma should also be appended after the last specified bit rate. After which, you should append ".f4m" to the end of the URL. A sample CDN URL is provided below.

```
http://wpc.0001.edgecastcdn.net/020001/videos/fly,110,400,650,.mp4.f4m
```

Note: If the filename for the desired set of videos contain a suffix, then you may append it after the last comma (e.g., `fly,110,400,650,Kbps.mp4.f4m`).

Note: For additional information on how to set the player URL, please refer to the **Single vs. Multiple Streams** section below.

Note: For information on how to upload content to CDN storage, please refer to the **CDN Origin Servers** section in the **Configuring an Origin Server** chapter of the **HTTP Large Administration Guide**.

Single vs. Multiple Streams

A media player can request a stream generated from a single H.264 asset or it can analyze a user's environment at frequent intervals and dynamically choose from a set of streams of varying quality the one that will provide the best user experience. The number of H.264 assets referenced in the player URL determines the media player's behavior.

Single Stream

Stream a single H.264 asset by modifying the CDN URL that points to it as indicated below.

1. Change the origin identifier from "00" to "02."
2. Append "f4m" after the file name extension.

Point the desired media player to the updated CDN URL. The proper syntax for the player URL is defined below.

Player URL Format:

```
http://wpc.xxxx.edgecastcdn.net/02xxxx/path/filename.ext.f4m
```

Sample CDN URL:

```
http://wpc.0001.edgecastcdn.net/000001/videos/fly_100Kbps.mp4
```

Sample Player URL:

```
http://wpc.0001.edgecastcdn.net/020001/videos/fly_100Kbps.mp4.f4m
```

Multiple Streams

If multiple H.264 assets are referenced in the player URL, a media player will dynamically analyze a user's bandwidth and CPU usage and then stream the bit rate quality that will provide the optimal user experience. This capability requires the following:

- An H.264 asset for each desired bit rate stream should be stored in CDN storage. The location in CDN storage doesn't matter as long as they are all stored in the same directory.
- The filename for each H.264 asset should indicate its corresponding bit rate quality in Kbps (e.g., 100, 200, or 400). The bit rate quality is the only difference allowed in the file naming convention.
- Each bit rate stream should be indicated in the player URL.

Player URL Format

The player URL for multiple streams is similar to the one used for a single stream. The main difference between the two URLs is that the one for multiple streams must identify each asset from which a stream can be generated. Multiple assets can be specified within a single URL by following specific file naming conventions.

Keep the following items in mind when identifying different bit rates in a player URL.

- The terms *prefix* and *suffix* refer to the portions of the filename that appear before and after, respectively, the bit rate level.
- All filenames must have a common prefix (e.g., **video**100.mp4, **video**200.mp4, and **video**300.mp4).
- The end of the prefix should be indicated with a comma. This should be followed by the bit rate level for each desired stream as a comma-delimited list in the filename (e.g., video,100,200,300,.mp4).
- Each specified bit rate stream should only include the bit rate quality in Kbps. Including any other data or using different units will generate a malformed playlist.
- A suffix identifies the portion of the filename that extends beyond the bit rate level (e.g., video100**kbps**.mp4, video200**kbps**.mp4, and video300**kbps**.mp4). If there is a suffix, then you may append it after the last comma (e.g., video,100,200,300,kbps.mp4).

The following CDN URL illustrates the proper syntax for filenames that contain a suffix:

```
http://wpc.xxxx.edgecastcdn.net/02xxxx/path/prefix,bitrate1,bitrate2,bitrateN,suffix.ext.f4m
```

The following CDN URL illustrates the proper syntax for filenames that do not contain a suffix:

```
http://wpc.xxxx.edgecastcdn.net/02xxxx/path/prefix,bitrate1,bitrate2,bitrateN,.ext.f4m
```

Example

This sample scenario illustrates how to construct a player URL that can reference multiple H.264 assets.

H.264 Assets (CDN URLs)

```
http://wpc.0001.edgecastcdn.net/000001/videos/fly_100Kbps.mp4
```

```
http://wpc.0001.edgecastcdn.net/000001/videos/fly_200Kbps.mp4
```

```
http://wpc.0001.edgecastcdn.net/000001/videos/fly_300Kbps.mp4
```

Sample Player URL:

```
http://wpc.0001.edgecastcdn.net/020001/videos/fly_,100,200,300,Kbps.mp4.f4m
```

Purging Live & On-Demand Content

Live and on-demand content can be purged from our network. A purge request will remove the cached version of the live or on-demand content from our edge servers.

Below are a couple of scenarios under which you may wish to purge content:

- **Live Event:** A purge request can be a quick and easy way to prevent users from viewing cached portions of your live stream after your live event has completed.
- **On-Demand Content:** Purge this type of content when you would like to distribute updated content without modifying your existing links. Simply update the source media on the origin server and then purge the cached content. This will force our edge servers to forward requests for that content to your origin server.

This streaming solution generates the following assets for each requested live or on-demand stream:

- General and bit rate-specific manifest files
- Bit-rate specific fragments

This means that purging the player URL will not purge the corresponding live or on-demand content. In order to properly purge this content, you will need to perform one of the following:

- Purge the parent folder:

Live Event:

`http://wpc.xxxx.edgecastcdn.net/22xxxx/InstanceName/EventName/*`

On-Demand Content:

`http://wpc.xxxx.edgecastcdn.net/02xxxx/path/*`

- Purge all variations of the base file name:

On-Demand Content:

`http://wpc.xxxx.edgecastcdn.net/02xxxx/path/basefileName*`

Keep the following information in mind when purging content:

- The recommended approach for purging varies by stream method:
 - **Live Event:** Purge the parent folder.
 - **On-Demand Content:** Purge all variations of the base file name. This will ensure that other content stored in the parent folder will not be inadvertently purged.
- It is important to specify a CDN URL that points to HDS (i.e., 22 or 02). Otherwise, the referenced content will not be purged.

- If you plan on purging a live event, keep in mind that the CDN URL used to play back the stream will not match the purge URL. A sample CDN and purge URL is provided below.

Sample Player URL (CDN URL):

```
http://wpc.0001.edgecastcdn.net/hds-live/200001/default/event1/mystream.f4m
```

Sample Purge URL:

```
http://wpc.0001.edgecastcdn.net/220001/default/event1/*
```

- The purge instructions defined in this topic apply regardless of whether the player URL references a single or multiple H.264 assets.

Purging a Live Event Example

Provided below is an example of how to purge a live event.

Sample Player URL:

```
http://wpc.0001.edgecastcdn.net/hds-live/200001/default/myevent1/fly,110,400,650,.f4m
```

Purge the parent folder:

```
http://wpc.0001.edgecastcdn.net/220001/default/myevent1/*
```

Purging On-Demand Content Example

Provided below are examples of two different methods by which on-demand content can be purged.

Sample Player URL:

```
http://wpc.0001.edgecastcdn.net/020001/videos/fly,110,400,650,.mp4.f4m
```

Purge the parent folder:

```
http://wpc.0001.edgecastcdn.net/020001/videos/*
```

Purge all variations of the base file name:

```
http://wpc.0001.edgecastcdn.net/020001/videos/fly*
```