

EdgeCast Networks, Inc.

Flash Media Streaming Administration Guide



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.

ADOBE, FLASH MEDIA LIVE ENCODER, and FLASH 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 is a registered trademark of Microsoft Corporation.

About This Guide

Flash Media Streaming Administration Guide

Version 3.61

6/25/2012

© EdgeCast Networks, Inc. All rights reserved.

Table of Contents

Streaming Flash Media	1
Overview	1
Available Options	1
Live StreamCast.....	2
Overview	2
Live StreamCast Requirements	4
Setting Up a Basic Live StreamCast.....	4
Encoder	4
Flash Player	6
Sample Flash Player Code	7
Archiving Your Stream	7
Testing Your Live StreamCast.....	9
Dynamic Stream Switching	10
Setting Up a Multi-bit Rate Live StreamCast.....	10
On-Demand Streaming (RTMP)	13
Overview	13
On-Demand Requirements	15
Setting Up Video On-Demand.....	16
Storage Location	16
Flash Player	17
Sample Flash Player Code	17
Loading On-Demand Content	18
Automatically Loading Assets through the Web Services REST API	19
Purging On-Demand Content.....	19
Automated Purging Through the Web Services REST API.....	19
Manually Purging Assets	20

Testing Video On-Demand.....	22
Dynamic Stream Switching	24
Setting Up Multi-bit Rate On-Demand Content.....	24
Configuring an Origin Server.....	26
Overview	26
Customer Origin Server.....	26
Hostnames/IP Addresses	27
HTTP Host Header	28
Load Balancing	28
Customer Origin Management	29
Content Expiration Headers.....	31
CDN Origin Servers.....	31
Masking the URL of a CDN Origin Server (CNAME)	32
Securing Your Flash Media.....	34
Overview	34
Preventing the Unauthorized Viewing of Your Flash Media.....	34
Using Tokens to Secure Flash Media	34
Authenticating Your SWF File	37
Encrypting Your Stream (RTMPE)	38
Preventing the Unauthorized Publishing of Flash Media	38
Publishing a Live Stream	40
Appendix A.....	41
Migrating from Live Streaming to Live StreamCast.....	41
Updating your Encoder's Configuration.....	41
Publishing Point URL	41
Stream Name	41
Updating your Flash Player Implementation	42
Updating HTTP Client Links to your Stream.....	42
Announcing your Stream to our Servers.....	42
Appendix B.....	43
Code Samples.....	43
Live StreamCast.....	43

JW Player Code Sample for Live StreamCast	43
Flowplayer Code Sample for Live StreamCast	44
On-Demand.....	46
JW Player Code Sample for On-Demand Streaming	46
Flowplayer Code Sample for On-Demand Streaming	47
Dynamic Streaming (Multi-Bit Rate)	50
Setting Up Dynamic Streaming for JW Player	50
Setting Up Dynamic Streaming for SMIL Compliant Media Players	53
Glossary.....	55

Streaming Flash Media

Overview

Flash media can be streamed either through Flash Media Communication Protocol (RTMP) or HTTP protocol. This chapter will explain the differences between these two methods and the different types of stream configurations that you can employ.

Available Options

Flash media can be streamed through our network through any of the following methods:

Method	Platform	Description
Live StreamCast	Flash Media Streaming	Feature overview: <ul style="list-style-type: none">• Stream a live event.• Uses Flash Media Communication Protocol (Real Time Messaging Protocol or RTMP).• Leverages security features.
On-Demand Streaming	Flash Media Streaming	Feature overview: <ul style="list-style-type: none">• Stream on-demand content.• Uses Flash Media Communication Protocol (Real Time Messaging Protocol or RTMP).• Leverages security features.
HTTP Progressive Download	HTTP Large Object	Feature overview: <ul style="list-style-type: none">• Stream on-demand content.• Easy to implement.• Uses standard HTTP protocol which is widely supported through firewalls.• Leverages a resilient infrastructure.

Note: For more information on HTTP Progressive Download, please refer to the **HTTP Large Object Administration Guide**.

Live StreamCast

Overview

Live StreamCast provides the capability to efficiently stream live video to your clients. The following diagram demonstrates the process through which a live video stream is transmitted to your clients.

Note: For the purpose of simplifying this example, only the North American portion of our worldwide network is displayed below.



How Live Video Streams to Your Client

The above diagram demonstrates four different steps that take place when streaming live video to your clients. These steps are:

1. An encoder encodes a video and makes it available to a publishing point.
2. The server associated with the publishing point broadcasts the live stream so that it can be pulled from any of our worldwide points-of-presence (POPs).
3. When requested by a user, the stream is pulled from the publishing point to the POP closest to each request.
4. All requests for that live stream from a particular region are served by a single POP. The live stream is delivered from the POP to the client through his/her ISP.

Encoding Video

The first step for serving a live stream occurs when your encoder encodes your live video and makes it available to a publishing point. We recommend that you choose the publishing point closest to the computer hosting your encoder. This will ensure optimal data transfer speed between your encoder and our network. In the above diagram, the closest publishing point to the encoder is North America, West Coast United States.

Live Stream Broadcast

Once the encoded video has been pushed to our publishing point, it is ready to be broadcast as a live stream by our ingest server. Our publishing points have been strategically placed in very close proximity to our POPs. This allows your live streams to immediately take advantage of our worldwide network. In turn, this ensures optimal data transfer speed between the publishing point and our POPs. In the above diagram, the closest POP to the publishing point is located in Los Angeles.

Pulling the Live Stream

Once a live stream is being broadcast, it is ready to be distributed throughout our worldwide network upon the request of one of your clients. The North American portion of our worldwide network is represented by orange lines in the above diagram. When a client requests a live stream, the POP closest to your client will retrieve the stream from the designated publishing point. In the above diagram, there are four clients trying to view the live stream. These clients are located close to Seattle and Atlanta. The Seattle and Atlanta clients will receive the live stream through the Seattle POP and Atlanta POP, respectively. By sending the stream through our network and bypassing traditional Internet communication routes, we can ensure the efficient transmission of your stream and reduced bandwidth load on your network.

Live Stream Delivery

The last leg involved in the delivery of your live stream requires that your client's ISP deliver the live stream from our POP. This is indicated by a purple line in the above diagram. Each of your clients will then use a Flash player to enjoy uninterrupted live streaming.

When a client requests a live video stream, a check is performed to find out whether another client in the same region is currently viewing it. If it is currently being viewed in the same region, then that client will instantly be able to take advantage of the live stream.

Securing Live Streams

This is the basic setup for streaming a live video feed through our worldwide network. In addition to simply improving the performance of your live video feeds, you can protect them in many different ways. The protection that we offer prevents the illegal modification or viewing of your video feeds. For more information on how you can protect your live streams, please refer to the **Securing Your Flash Media** chapter.

Live StreamCast Requirements

Live StreamCast requires an encoder, access to the content delivery network, and a Flash player. The requirements for each of these items are listed below.

System Requirements:

- **Encoder:** Live StreamCast has been tested with Adobe Flash Media Live Encoder 3.0 and higher. The system requirements for this computer are determined by your encoder and the number of stream instances it will broadcast. For more information on system requirements or the configuration of your encoder, please refer to Adobe's web site.
- **Content Delivery Network:** We provide publishing servers that will ingest your live stream and transmit it to all necessary edge servers for delivery to your end-users.
- **Flash Player:** Our service is compatible with any Flash player that supports the FCSUBSCRIBE call and can handle a stream encoded by Flash Media Live Encoder 3.0 or higher.

Note: To find out whether a Flash player supports the FCSUBSCRIBE call, please contact the software manufacturer of your desired player.

Note: Our testing indicates that the following versions of JW Player work with Live StreamCast: versions 4.6, 5.1 and above. However, it is important to note that we do not provide Flash player support. If you experience issues with your Flash player implementation, please contact that player's software manufacturer for technical support.

Setting Up a Basic Live StreamCast

There are two main elements to a basic configuration of Live StreamCast. These elements are an encoder and the Flash client. We will discuss each one of these elements below.

Encoder

Live StreamCast requires that you set up a computer where the feed will be encoded with Flash Media Live Encoder (FMLE) 3.0 or higher. Configuring Flash Media Live Encoder requires that you perform the following tasks:

- Select and configure the audio/video device that contains the desired feed.
- Determine the bit rate and output size of the video feed.
- Configure how the encoder streams to our Flash Media Server. This configuration involves specifying a publishing point URL and the stream name. Both of these options are explained below.

Publishing Point URL (FMS URL)

Before an encoder can send an encoded feed to our Flash Media Streaming servers, you will need to indicate where to publish your live stream. You should specify the location of the publishing point closest to your encoder.

The following table provides the URLs for the available Live StreamCast publishing points.

Publishing Point	URL
North America: East Coast U.S.	rtmp://fso.dca.xxxx.edgecastcdn.net/20xxxx/Path
North America: West Coast U.S.	rtmp://fso.lax.xxxx.edgecastcdn.net/20xxxx/Path
Europe: Amsterdam	rtmp://fso.ams.xxxx.edgecastcdn.net/20xxxx/Path
Europe: Frankfurt	rtmp://fso.fra.xxxx.edgecastcdn.net/20xxxx/Path
	Note: This ingest location is only available on FMS 4.5. Please contact your CDN account manager for more information.
Australia	rtmp://fso.syd.xxxx.edgecastcdn.net/20xxxx/Path

Tip: The publishing point URL should not end with a forward slash (/).

Tip: We support the use of a backup publishing point. If one publishing point should experience technical issues, a backup publishing point would take over the task of publishing the stream. In Adobe Flash Media Live Encoder 3.1 and above, this option can be set through the **Backup URL** option.

Note: You should replace *xxxx* with your CDN account number. Your CDN account number can be found in the upper-right hand corner of the MCC. You should also replace *Path* with the path to the folder where you would like to publish your stream.

The folder where you will publish your live stream is important for the following two reasons.

- **EC360 Analytics Suite:** One of our reports allows you to view bandwidth and client usage by folder. If you publish certain types of streams to different folders, then you can view statistics by stream type.
- **Stream Archival:** The Server-side archiving feature relies on the specified path to determine where a live stream will be archived on CDN storage. If you publish to the root folder, then archived streams will be stored in the "_definst_" folder. This folder can be found directly off the root folder of the CDN origin server.

Tip: Publishing to a folder that does not exist will not impair CDN functionality (e.g., EC360 Analytics Suite, Live Authentication Key, Token-Based Authentication, etc.). However, the folder will not be created on CDN storage unless the Server-side archiving feature is leveraged.

Stream Name

Another essential configuration option is the name assigned to the stream generated by an encoder. This name must include a security token, which is known as a Live Authentication key. A Live Authentication key ensures that only you can publish streams from your account. The proper format for specifying a stream name in your encoder is:

- *StreamName? LiveAuthenticationKey*

For example, if you decide that you would like to call your stream "MyStream" and your global Live Authentication key is "MyPrivateKey," then your actual stream name would be "MyStream?MyPrivateKey."

Important: An encoder will not be permitted to stream content through your account if a Live Authentication key has not been specified as a part of the stream name.

Note: For more information on how Live Authentication keys work, please refer to the **Preventing the Unauthorized Publishing of Flash Media** section below.

Note: If the Server-side archiving feature is currently enabled, then the stream name determines the base name of the file that will be stored on the server and the video format that will be used. For more information, please refer to the **Archiving Your Stream** section.

Flash Player

Live StreamCast requires that you specify three different elements when implementing your Flash player. These elements are a player URL, a stream name, and FCSubscribe.

- **Player URL:** The player URL is the path to the stream on our CDN server. The notation for this path is provided below. Please remember to replace *xxxx* with your CDN account number. You should also replace *Path* with the path to the folder to which your encoder is publishing.
 - `rtmp://fml.xxxx.edgecastcdn.net/20xxxx/Path`
- **Stream Name:** The stream name is the name of the stream, as defined in the encoder. Make sure that you do not include the Live Authentication key. You should also not include a question mark as a part of the stream name unless you are taking advantage of Token-Based Authentication.
 - **Token-Based Authentication:** If the stream is protected by Token-Based Authentication, then you will need to append a question mark (?) followed by the token generated from the primary encryption key. For example, if your stream name is "WebCam01" and the token corresponding to your primary encryption key is "a4fbc3710fd3449a7c99984d1b86603c22be1006d830b," then you would specify "WebCam01?a4fbc3710fd3449a7c99984d1b86603c22be1006d830b" as your stream name. For detailed information, please refer to the **Token-Based**

Authentication User Guide, which is available from the Media Control Center (MCC).

- **FSubscribe:** FSubscribe is a call used to determine when a connection starts or stops. You will need to set FSubscribe to true when implementing your Flash player. The manner in which you can set this method varies according to the Flash player that you are using. To find out whether a Flash player supports FSubscribe or how to set it, please contact the software manufacturer of your desired player.

Important: If a stream is protected by Token-Based Authentication, then the player URL's query string should consist solely of the token value. Additional query string parameters are unsupported.

Tip: The player URL can also be found on the [Flash Streaming](#) page of the **Flash** tab in the MCC.

Reminder: The default port used for the RTMP and RTMPE protocols is 1935. If your client's Flash player does not allow communication through port 1935 and you have enabled tunneling, then you will be able to send your stream through the HTTP protocol using port 80.

Sample Flash Player Code

Sample code can be viewed from the **Live StreamCast** section of **Appendix B**.

Archiving Your Stream

A Flash stream can be recorded on a CDN origin server as it is being streamed. One use of this feature is to provide on-demand capabilities for Flash content that was originally streamed live.

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 to a path that replicates the origin path specified in the publishing point URL. For example, if you set your publishing point to:

```
rtmp://fso.lax.xxxx.edgecastcdn.net/20xxxx/Videos/July/2012
```

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

```
/Videos/July/2012
```

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 Management** page, which can be found on the **Media Manager** tab.

File Format

By default, Flash streams are saved as FLV files. However, you have the option to save them using either the H.264 F4V or MP4 file format. If you prefer to use a different file format, then you will need to use the following format when setting the stream name in your preferred encoder:

- `mp4:StreamName.f4v?LiveAuthenticationKey`
- `mp4:StreamName.mp4?LiveAuthenticationKey`

Note: If you plan on encoding your video using the H.264 format, then you should make sure that you only archive files using the F4V or MP4 file format. Archiving H.264 encoded video as an FLV file may cause data loss.

Note: The stream naming convention (e.g., `mp4:StreamName.mp4`) defined above should also be used when defining the stream name in the media player.

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 one of the following:

- **Append (enabled):** `_StreamName.append.ext`
- **Append (disabled) – Overwrite Mode:** `StreamName.ext`

The term *ext* refers to the filename extension that will be assigned to your archived file. This can either be FLV, F4V, or MP4, depending on the type of file format used to archive your Flash media.

If the specified ingest server is unable to resume encoding your video 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 conventions for files archived under these circumstances are provided below.

- **Append (enabled):** `_StreamName.append.#.ext` (e.g., `Demo.append.1.flv`)
- **Append (disabled) - Overwrite:** `StreamName.#.ext` (e.g., `Demo.1.flv`)

Testing Your Live StreamCast

A live stream can be viewed without having to embed a video player onto a web page. This allows you to quickly verify that your live stream has been configured properly. Before you can test out your live stream, you will need to meet the following requirements:

- Make sure that your encoder is properly pushing encoded video to a publishing point. This involves verifying the publishing point location and the live authentication key.
- Download the latest version of JW Player. Upload "Player.swf" to a CDN or customer origin server.

Once you have met the above requirements, then you should simply type the following URL in your browser's address bar:

```
http://wpc.xxxx.edgecastcdn.net/PlayerPath/player.swf?streamer=rtmp://fml.xxxx.edgecastcdn.net/20xxxx/StreamPath&type=rtmp&rtmp.subscribe=true&file=StreamName.flv
```

Note: The above sample URL uses the HTTP Large Object platform for the SWF file. Alternatively, you may make the Flash player available through the HTTP Small Object platform. You can do so by replacing "wpc" with "wac" in the above sample URL.

You will need to replace the following items in the URL:

- **xxxx:** This should be replaced with your CDN account number. Your CDN account number can be found in the upper-right hand corner of the MCC.
- **PlayerPath:** This term should be replaced by the path to the folder where "Player.swf" is stored.
 - **CDN URL:** If you are not using an edge CNAME, then this path should start with the content access point (i.e., *yyxxxx/Path*). The variable "yy" should be replaced with the appropriate origin identifier. If you are using a CDN origin server, then yy should be set to "00." If you are using a customer origin server, then it should be set to "80."
 - **Edge CNAME URL:** This path should start with the name assigned to the appropriate edge CNAME followed by the path to the Flash player folder. It should not include the content access point.
- **StreamPath:** This term should be replaced by the path to the publishing point location. Make sure that this path does not end with a forward slash (/).
- **StreamName:** This term should be replaced by the name assigned to the stream by your encoder.

Important: CDN and edge CNAME URLs are case-sensitive.

Dynamic Stream Switching

A basic Live StreamCast configuration requires that you decide on a single bit rate that will provide the optimal viewing experience to your clients. This means that you have to balance video quality with your clients' CPU and bandwidth limitations. Configuring your media player to use dynamic streaming for your live event, on the other hand, allows you to provide video quality that satisfies your high-end users without diminishing the viewing experience for users with limited bandwidth. It is able to do this by providing multiple streams for a live event. Each stream contains video with varying degrees of quality. A Flash player can then choose the stream that best suits each user. Typically, a user's bandwidth and CPU usage are used to make this determination.

Note: Whether a Flash player supports multi-bit rate streams and the criteria that it uses to determine the stream quality that will be requested depends on that player's manufacturer.

Setting Up a Multi-bit Rate Live StreamCast

A multi-bit rate Live StreamCast can be configured in a similar way to a basic Live StreamCast. The major configuration differences are listed below.

- An encoder must be configured to generate a stream for each desired bit rate. A live authentication key must be specified for each stream.
 - If you plan on using stream keys, then you will need to create a stream key for each stream. For more information, please refer to the **Live Authentication and Dynamic Streaming (Multi-Bit Rate Streams)** section in the **Preventing the Unauthorized Publishing of Flash Media** topic.
- The available streams for a live event must be listed in a multi-bit rate configuration file (e.g., SMIL or XML).
- A Flash player will need to point to a multi-bit rate configuration file (e.g., SMIL or XML) instead of directly to the live stream.

Tip: An optional configuration difference is to increase the number of keyframes that will be generated for your streams. Keyframes are a crucial factor in determining when a Flash player can switch to a different quality stream. Using a small keyframe interval (e.g., 2 to 4 seconds) reduces the chance that a player will buffer video or be affected by playback stutter.

Note: Once you have set up dynamic streaming, you will simply need to start encoding your live event. Your Flash players will automatically request the appropriate bit rate stream and adjust the bit rate level to handle changing network conditions.

Choosing Bit Rate Levels

The most important step in configuring your multi-bit rate Live StreamCast is determining how many and the quality of the streams that you will make available to your clients. Some general guidelines on how to choose bit rate quality are provided below.

- **Low quality:** This stream should not require much in terms of bandwidth or CPU usage. All of your clients should be able to view this type of stream without experiencing delays due to buffering, disconnects, or playback stutter.
- **Balance between quality and bandwidth/CPU requirements:** This type of stream will cater to most of your users, since it will be the highest quality video that they can view with their limited bandwidth or hardware.
- **High quality:** This stream should provide an optimal viewing experience to demanding clients that expect higher quality due to their available bandwidth and computer hardware.

Note: Keep in mind that the number of bit rate streams that can be generated by a single event may be limited by your encoder.

When configuring your encoder to generate a bit rate stream, you will need to select the desired bit rate and the resolution or output size of the video. The steps required to accomplish this depends on your encoder. For more information, please refer to the documentation provided with your encoder.

Note: Adobe Flash Media Live Encoder 3.1 and above provides the capability to create up to three bit rate streams. However, you must enable each desired bit rate stream by marking the checkbox that appears to the left of the corresponding bit rate settings.

Generating Multiple Bit Rate Streams

After you have chosen the quality of the bit rates that will be made available to your clients, you will need to inform the encoder that it needs to generate a stream for each one. This can be accomplished by using a variable in the stream name or by specifying a name for each desired bit rate stream. The steps required to accomplish this depends on your encoder. For more information, please refer to the documentation provided with your encoder.

If you are using Adobe Flash Media Live Encoder 3.1 and above, you can specify multiple stream names by performing one of the following:

- Use %i as a stream name variable (e.g., Stream%i?LiveAuthKey). The %i variable will be replaced by the number associated with each desired bit rate stream (e.g., Stream1, Stream2, and Stream3). This number can be viewed to the left of the settings used to configure the desired bit rate.
- Use a semi-colon (;) to separate each desired bit rate stream name. (e.g., Stream-300?LiveAuthKey;Stream-600?LiveAuthKey;Stream-900?LiveAuthKey).

Note: Keep in mind that the stream names defined above will need to be listed in the multi-bit rate configuration file.

Creating a Multi-bit Rate Configuration File

Once you have decided upon the names that will be assigned to each bit rate stream, you are ready to create your multi-bit rate configuration file. This file defines the location to which you are publishing your live stream and a name for each bit rate stream.

The standard multi-bit rate configuration file is a SMIL document. However, the type of configuration file that you should use depends on your Flash player. For example, JW Player uses an RSS document.

Sample code can be viewed from **Appendix B: Dynamic Streaming (Multi-Bit Rate)**.

Configuring Your Flash Player

The final step that you will need to take is to point your Flash player to a multi-bit rate configuration file instead of a stream.

Sample code can be viewed from **Appendix B: Dynamic Streaming (Multi-Bit Rate)**.

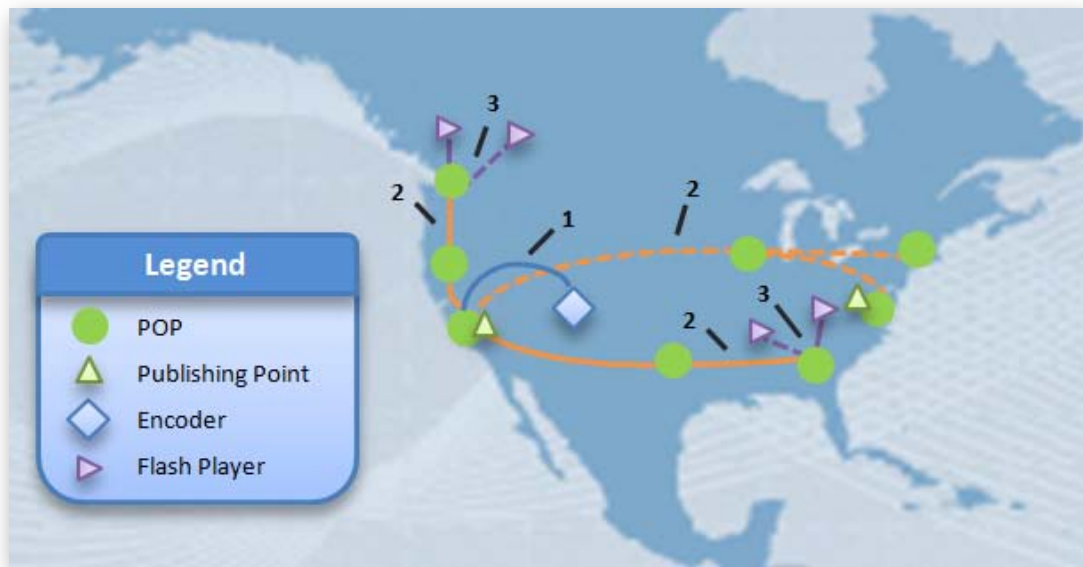
On-Demand Streaming (RTMP)

Overview

Our on-demand capabilities allow you to stream Flash media, including previously archived live streams, as your clients request them. The following diagram demonstrates the process through which an on-demand Flash stream is transmitted to your clients.

Note: An alternative way to stream on-demand Flash content is HTTP Progressive Download. For more information, please refer to the **HTTP Progressive Download** chapter in the **HTTP Large Object Administration Guide**.

Note: For the purpose of simplifying this example, only the North American portion of our worldwide network is displayed.



How On-Demand Content Streams to Your Client

The above diagram demonstrates three different steps that take place when streaming on-demand video to your clients. These steps are:

1. A Flash media file must be stored on a CDN or customer origin server. The first time that this Flash content is requested by a user it is retrieved from that origin server.
2. It is then served to the client that requested it and cached on the POP closest to that client.
3. Subsequent requests for cached content can be served directly from the POP.

Storing and Retrieving Flash Media

Before your on-demand content can be served to your clients, it must first be stored on a CDN or customer origin server. This step is essential since it allows the on-demand content to be requested through a CDN or edge CNAME URL. This type of URL allows your on-demand content to be streamed across our worldwide network.

On-demand content is streamed to your customers as they request them. This process involves streaming the content from the origin server to our worldwide network. In the above example, this step is represented by a blue line that goes from the origin server to a POP in Los Angeles.

Routing and Caching Flash Media

Once the requested Flash media is in our network, the next step involves sending the Flash media to the POP closest to the client requesting the content. Once the entire file has been served to a POP, it will be cached there. In the above diagram, there are four clients requesting on-demand content. These clients are located close to Seattle and Atlanta. The Seattle and Atlanta clients will receive the Flash media through the Seattle POP and Atlanta POP, respectively. This is represented by orange lines in the diagram. Sending Flash media through our network and bypassing traditional Internet communication routes ensures the efficient transmission of your Flash media and reduced bandwidth load on your network.

Flash Media Delivery

The last leg involved in the delivery of your on-demand content requires your client's ISP to deliver the Flash media stream from our POP. This is indicated by a purple line in the above diagram.

Keep in mind that once on-demand content has been cached on a POP, all future requests for that content can bypass retrieval from the origin server. When a client requests on-demand content, a check is performed to find out whether the requested Flash media has already been cached at the POP closest to your client. If it is already there, then the on-demand content is immediately served up to that client. In the above diagram, the dashed purple lines indicate the clients that will be able to enjoy even faster on-demand streaming.

On-Demand Requirements

On-Demand streaming requires an origin server, our content delivery network, and a Flash client. The requirements for each of these items are listed below.

System Requirements:

- **Origin server:** You can choose to store your Flash media files on either your own server (customer origin) or on a CDN storage server (CDN origin). If you choose to store your media files on your own server, then you will need to ensure that your server has sufficient bandwidth and storage space to handle all of your streaming needs. On the other hand, if you take advantage of our CDN origin servers, then you can rest assured that your files will stream from a server that has top of the line bandwidth and storage space.
 - **CDN Storage:** Make sure that the player URL does not point to the root folder of the CDN origin server. You must store your Flash content in another folder and specify the path to that folder when setting the player URL.
 - **Customer Origin:** Configure your web server to add either a Last-Modified (preferred) or an ETag header to each response. Either response header is required for caching purposes. If either response tag is not found, then requests for on-demand content will be served from your customer origin server. Additionally, you will need to create a customer origin configuration for the desired server.
- **Content Delivery Network:** We provide servers that transmit your on-demand content to all necessary edge servers for delivery to your end-users.
- **Flash Player:** Our service is compatible with any Flash player that can handle a stream encoded by Flash Media Live Encoder 3.0 or higher.

Reminder: The default port used for the RTMP and RTMPE protocols is 1935. If your client's Flash player does not allow communication through port 1935 and you have enabled tunneling, then you will be able to send your stream through the HTTP protocol using port 80.

Setting Up Video On-Demand

The capability to stream Flash media as your clients request them requires that you configure the storage location for your Flash media and upload the desired file(s). After which, you will need to implement your Flash player and point it to the desired Flash media file. We will discuss each of these items below.

Storage Location

We can retrieve Flash media from two different sources, which are CDN origin and customer origin servers. CDN origin refers to our set of storage servers. These servers have sufficient storage capacity to exceed your needs. Additionally, they have direct access to our CDN servers, which allows them to deliver your media files quickly. A customer origin server refers to any computer external to our network to which you have sufficient access to store and retrieve files. If you choose to store media files on your own server, then you will need to ensure that your customer origin server can communicate with our CDN network. This can be accomplished by creating a customer origin configuration and then allowing the IP addresses associated with our CDN servers access to your customer origin server through the HTTP protocol. For a complete list of IP blocks that should have access to your storage server, please refer to the [Customer Origin](#) page on the **Flash** tab of the MCC.

Note: For more information on the configuration required to set up a server for use with CDN service, please refer to the **Configuring an Origin Server** chapter.

Reminder: Do not store Flash media in the root folder of a CDN origin server. This type of configuration is not supported.

To store Flash media on a customer origin server

Use your preferred FTP client to authenticate to that server. Navigate to the desired path and then upload the desired Flash media files.

To store Flash media on a CDN origin server

Perform one of the following:

- **FTP Client:** Find out the FTP URL through which you can upload content by loading the [FTP](#) page from the **Media Manager** tab in the MCC. Use your preferred FTP client to log in to that server using your MCC log in information. Navigate to the desired path and then upload the desired Flash media files.
- **MCC:** From the [File Management](#) page on the **Media Manager** tab of the MCC, upload the desired Flash media file to the desired path on the origin server.

Flash Player

Once your Flash media files are hosted on a customer or CDN origin server, you are ready to set up your Flash player to stream on-demand content. This requires that you specify a player URL (stream) and the filename of the desired Flash media in your player code.

- **Player URL:** The player URL is the URL to the folder on either the customer origin or CDN origin server where your Flash media file is stored.
 - If you would like to stream content from CDN storage, then the URL to your on-demand content can be generated from the [Flash Streaming](#) page of the **Flash** tab in the MCC.
- **Filename:** In your embedded code, specify the filename including the extension (e.g., MyFlash.flv).
 - **Token-Based Authentication:** If the desired content is secured by Token-Based Authentication, then you will need to append a question mark (?) followed by the token generated from the primary encryption key. For example, if the media file is called "WebCam01.flv" and the token corresponding to your primary encryption key is "a4fbc3710fd3449a7c99984d1b86603c22be1006d830b," then you would specify "WebCam01.flv?a4fbc3710fd3449a7c99984d1b86603c22be1006d830b" as your stream name. For detailed information, please refer to the **Token-Based Authentication User Guide**, which is available from the Media Control Center (MCC).

Tip: If you do not wish to expose the CDN path to your clients, you may choose to create a mask for the URL of the CDN origin server using a CNAME. For more information, please refer to the **Masking the URL of a CDN Origin Server (CNAME)** section below.

Sample Flash Player Code

Sample code can be viewed from the **On-Demand** section of **Appendix B**.

Loading On-Demand Content

By default, our edge servers don't cache files until they are requested by a client. This means that the first customer in a particular region to request a file will cause the corresponding edge server to request the file from the origin server. After which, the file will remain cached on that edge server until it is automatically or manually purged. Therefore, the first customer to request a particular file will take slightly longer to retrieve that file than all other subsequent customers. If you have a large event or content that becomes simultaneously available to a large volume of end-users (e.g., a new movie release), then it might be helpful to load your Flash media content on our edge servers.

Note: There is a default limit of 50 concurrent load requests at any given time.

Note: For more information on purging on-demand content, please refer to the **Purging On-Demand Content** section below.

Note: Loading an asset will cause it to be cached across the CDN regardless of the CNAME used to reach it; provided that those CNAME records point to the same location.

Note: A URL containing a dollar sign symbol (i.e., \$) cannot be added to the load queue. The dollar sign symbol must be encoded as "%24" before such a URL can be added to the queue.

To load your on-demand Flash media to our edge servers

1. Navigate to the [My Edge](#) page, which can be found on the **Flash** tab of the MCC.
2. From the **Load URL** option, select the CDN URL for the server where your Flash media content can be found.
3. To the right of the **Load URL** option, type the full path to the desired Flash file that will be loaded.
 - Make sure that this path starts with a forward slash.
 - Make sure to include the filename extension when specifying the filename.
4. Click **Load**.

Note: Each load request needs to be processed. Therefore, your request will be placed in a queue. The **My Edge** page provides a means to keep track of when a request was made and when it was completed.

Automatically Loading Assets through the Web Services REST API

We provide an API that you can use to automatically load assets to all of our POPs. For detailed information on the method and the parameters that you should use, please consult the **EdgeCast Web Services REST API** guide. This guide is available from the home page of the MCC.

Reminder: The Web Services REST API allows you to quickly send multiple requests to load content on our edge servers. However, you should keep in mind that there is a default limit of 50 concurrent load requests at any given time.

Purging On-Demand Content

Purging an asset deletes the cached version of the asset from all of our edge servers. This prevents an old version of an asset from being delivered to your clients. This is useful when you would like to ensure the delivery of a new version of the asset in question.

Important: Purging is not the same thing as deleting content from the origin server. Purging an asset will not delete the source asset. It will simply remove the cached instance of that asset from all of our edge servers. If you would like to remove the original asset, then you will need to delete it from the origin server through your preferred FTP client.

Note: There is a default limit of 50 concurrent purge requests at any given time.

Note: A URL containing a dollar sign symbol (i.e., \$) cannot be added to the purge queue. The dollar sign symbol must be encoded as "%24" before such a URL can be added to the queue.

Automated Purging Through the Web Services REST API

We provide a Web Services REST API that you can use to automatically purge content and/or folders from all of our POPs. For detailed information on the method and the parameters that you should use, please consult the **EdgeCast Web Services REST API** guide. This guide is available from the home page of the MCC.

Reminder: The Web Services REST API allows you to quickly send multiple purge requests. However, you should keep in mind that there is a default limit of 50 concurrent purge requests at any given time.

Manually Purging Assets

Cached Flash media files can be purged from our edge servers through the **My Edge** page, which can be found on the **Flash** tab of the MCC. Keep in mind that you may purge a specific asset, a set of assets, the entire contents of a folder, or recursively.

Tip: If you have assets that are updated frequently, then you should consider using a naming convention that assigns unique names to your new assets. This will ensure that future requests will go to the new assets without the need for purging the old ones.

Tip: An alternative to changing your file naming convention for assets that are updated frequently is to use the **Bulk Purge** option. This option allows you to specify a list of assets that will be placed into the purge queue. If you keep a record of frequently purged assets, then you can copy and paste them into this option. Please keep in mind that there is a default limit of 50 concurrent purge requests at any given time.

Note: An asset only needs to be purged a single time per location. The asset will be purged regardless of the CNAME used to reach it; provided that those CNAME records point to the same location.

To manually purge assets

1. Navigate to the [My Edge](#) page, which can be found on the **Flash** tab of the MCC.
2. Under the **Purge From Edge** section, select the location of the asset(s) that you would like to purge from the **Purge URL** option.
3. To the right of the **Purge URL** option, type the path to the folder containing the asset(s) that you would like to purge. Make sure that the path starts with a forward slash (/).
4. Perform one of the following:
 - **To purge a specific asset:** Append the name of the asset that you would like to purge to the relative path specified in step 3.
 - **To purge a set of assets:** Determine the naming convention and/or file type that will be used to identify the assets that will be purged. Append that pattern to the relative path specified in step 3 (e.g., /2012/July*.fl*).
 - **To purge a folder's content:** Append a forward slash, an asterisk (i.e., /*), a period, and another asterisk to the relative path specified in step 3 (e.g., /2012/*.*).
 - **To recursively purge a folder's content:** Make sure that the path points to the folder whose contents you would like to purge and that a forward slash and an asterisk (i.e., /*) follow the name of the desired folder.
5. Click **Purge**.

Important: The **Purge** button applies to both the **Purge URL** and the **Bulk Purge** option. However, the **Bulk Purge** option takes precedence over the **Purge URL** option. This means that the **Purge URL** option will be ignored when the **Bulk Purge** option contains a value.

Important: An asterisk (*) can either be used as a wildcard in a filename pattern or to recursively purge a folder's content. However, it cannot be used as a wildcard when specifying a directory. Additionally, you should keep in mind a recursive purge is always performed for purge requests that end with a forward slash followed by an asterisk (/*). If you would like to purge a folder's content, append /*.* to the desired purge directory.

Note: Each purge request needs to be processed. Therefore, your request will be placed in a queue. The **My Edge** page provides a means to keep track of when a request was made and when it was completed.

To perform a bulk purge

1. Navigate to the **My Edge** page, which can be found on the **Flash** tab of the MCC.
2. In the **Bulk Purge** option, type the CDN or edge CNAME URL that points to the asset that will be purged.
3. If you would like to purge another asset, press ENTER and then specify the desired CDN or edge CNAME URL. Repeat this step as needed.
4. Click **Purge**.

Note: Each CDN or edge CNAME URL specified in the **Bulk Purge** option must be placed on a separate line. This can be accomplished by using a carriage return to delimit each CDN or edge CNAME URL. Using any other type of character (e.g., a comma) as a delimiter may prevent the asset from being purged.

Purging an Individual Asset

A purge can be performed on an individual asset by specifying the filename of the desired asset. The following table indicates the syntax that should be used to purge an individual asset.

Syntax	Description
<i>/FolderPath/Asset.ext</i>	Purging a specific asset can be performed by simply specifying the relative path to the directory where it can be found and the filename of the desired asset (e.g., Video.flv).

Purging a Set of Assets

A pattern can be used to determine the assets that will be purged from the specified directory. This can be accomplished through the use of one or more asterisks. Each asterisk represents one or more characters. If you would like to purge a particular file type, then you can simply append an asterisk, a period, and the desired filename extension to the directory where the purge will take place (e.g., /*.flv).

A list of sample purges is provided below.

Sample Syntax	Description
/Folder01/*.*	This sample purge syntax will purge the entire contents of the directory called "Folder01." This purge request will not be performed recursively.
/Folder01/*.flv	This sample purge syntax will purge all FLV assets from the directory called "Folder01."
/Folder01/a*.fl*	This sample purge syntax will purge all assets that start with the letter "a" and whose filename extension starts with "fl" (e.g., activity.flv).

Recursive Purging

We have already seen how to purge the contents of a folder through the use of wildcards (i.e., *.*). If you would also like to purge all of the subfolders of the specified directory, then you should use the following syntax instead:

- */FolderPath/**

The term *FolderPath* represents the relative path to the location where the directory can be found. Including a single asterisk in the syntax indicates that a recursive purge should be performed on that location.

Note: An asterisk cannot be used to specify a folder pattern. It can only be used as specified above or when defining a pattern that will be used to define the set of assets that will be purged (e.g., */*.flv*).

Testing Video On-Demand

Flash media can be viewed without having to embed a Flash player onto a web page. This allows you to quickly verify that your player URL is correct. Before you can test your on-demand content, you will need to meet the following requirements:

- Make sure that your Flash media is stored on a CDN or customer origin server.
- Download the latest version of JW Player. Upload "Player.swf" to a CDN or customer origin server.

Once you have met the above requirements, then you should simply type the following URL in your browser's address bar:

```
http://wpc.xxxx.edgecastcdn.net/PlayerPath/player.swf?streamer=rtmp://fms.xxxx.edgecastcdn.net/FlashPath&type=rtmp&file=Filename.flv
```

You will need to replace the following items in the URL:

- **xxxx:** This should be replaced with your CDN account number. Your CDN account number can be found in the upper-right hand corner of the MCC.
- **PlayerPath:** This term should be replaced by the path to the folder where "Player.swf" is stored.
 - **CDN URL:** If you are not using an edge CNAME, then this path should start with the content access point (i.e., *yyxxx/Path*). The variable "yy" should be replaced with the appropriate origin identifier. If you are using a CDN origin server, then yy should be set to "00." If you are using a customer origin server, then it should be set to "80."
 - **Edge CNAME URL:** This path should start with the name assigned to the appropriate edge CNAME followed by the path to the Flash player folder. It should not include the content access point.
- **FlashPath:** This term should be replaced by the path to the folder where your Flash media content is stored. Make sure that this path does not end with a forward slash (/).
 - **CDN URL:** If you are not using an edge CNAME, then this path should start with the content access point (i.e., *yyxxx/Path*). The variable "yy" should be replaced with the appropriate origin identifier. If you are using a CDN origin server, then yy should be set to "00." If you are using a customer origin server, then it should be set to "80."
 - **Edge CNAME URL:** This path should start with the name assigned to the appropriate edge CNAME followed by the path to the Flash content folder. It should not include the content access point.
- **Filename:** This term should be replaced by the filename of the desired Flash media content.

Important: CDN and edge CNAME URLs are case-sensitive.

Note: If you have protected the desired Flash content with Token-Based Authentication, then you will need to append a token value.

Note: The above sample URL uses the HTTP Large Object platform for the SWF file. Alternatively, you may make the Flash player available through the HTTP Small Object platform. You can do so by replacing "wpc" with "wac" in the above sample URL.

Dynamic Stream Switching

The amount of bandwidth that can be used by a client to view streaming content is variable due to many factors, such as their current network (e.g., home vs. office vs. coffee shop) and playback device (e.g., smart phone vs. desktop). As a result, you may wish to provide varying quality levels of the same video to your clients. Typically, this would be performed by creating several links to the desired content and then allowing a user to select the video quality that will produce the best viewing results. A major drawback of this type of configuration is that it effectively locks a user into the selected video feed, since switching to a different quality level would require the user to restart that video.

Multi-bit rate functionality changes your clients' viewing experience by switching the burden of choosing an appropriate video quality from the user to the Flash player. It allows the Flash player to automatically adjust the quality of the video being displayed according to current network conditions and CPU usage. It is able to do this by continually monitoring a client's environment, assessing the best bit rate for each client, and requesting a higher or lower bit rate Flash video according to the information provided in a configuration file.

Note: A Flash player will not immediately switch to a different bit rate when it detects a significant change in a client's network conditions and/or CPU usage. Instead, it will wait until it detects the next key frame.

Note: Whether a Flash player supports multi-bit rate streams and the criteria that it uses to determine the stream quality that will be requested depends on that player's manufacturer.

Setting Up Multi-bit Rate On-Demand Content

Multi-bit rate functionality can be implemented in much the same manner as on-demand content. The major configuration differences are listed below.

- Multiple video files with varying bit rate levels will be provided instead of a single video file.
- The available video files must be listed in a multi-bit rate configuration file. This file can either take advantage of SMIL or RSS syntax. The syntax that should be used depends on your Flash player.
- A Flash player will need to point to a multi-bit rate configuration file instead of directly to the video file.

Tip: It is recommended that you encode your videos with a high frequency of key frames. Key frames are a crucial factor in determining when a Flash player can switch to a different quality stream. Using a small key frame interval (e.g., 2 to 4 seconds) reduces the chance that a player will buffer video or be affected by playback stutter.

Choosing Bit Rate Levels

The most important step for adding multi-bit rate functionality to your on-demand content is to determine how many and the quality of the video files that you will make available to your clients. Some general guidelines on how to choose bit rate quality are provided below.

- **Low quality:** This type of video file should not require much in terms of bandwidth or CPU usage. All of your clients should be able to view this type of stream without experiencing delays due to buffering, disconnects, or playback stutter.
- **Balance between quality and bandwidth/CPU requirements:** This type of video file will cater to most of your users, since it will be the highest quality video that they can view with their limited bandwidth or hardware. Most of your video files should fall under this category.
- **High quality:** This type of video file should provide an optimal viewing experience to demanding clients that expect higher quality due to their available bandwidth and computer hardware.

Note: Video files of varying quality can be either be created when streaming a live event or from an existing video file. When configuring your encoder, you will need to select the desired bit rate and the resolution or output size of the video. The steps required to accomplish this depends on your encoder. For more information, please refer to the documentation provided with your encoder.

Note: Adobe Flash Media Live Encoder 3.1 and above provides the capability to create up to three bit rate streams from a live event. This capability requires that you enable each desired bit rate stream by marking the checkbox that appears to the left of the corresponding bit rate settings.

Creating a Multi-bit Rate Configuration File

Once you have generated the desired bit rate video files, you are ready to create a multi-bit rate configuration file. This file provides a listing of video files and the minimum bit rate level required before a Flash player can switch to it.

The standard multi-bit rate configuration file is a SMIL document. However, the type of configuration file that you should create depends on your Flash player. For example, JW Player only supports multi-bit rate configuration through an RSS document.

Sample code can be viewed from **Appendix B: Dynamic Streaming (Multi-Bit Rate)**.

Configuring Your Flash Player

The final step that you will need to take is to point your Flash player to a multi-bit rate configuration file instead of a single video file. This can be accomplished through the "file" variable.

Sample code can be viewed from **Appendix B: Dynamic Streaming (Multi-Bit Rate)**.

Configuring an Origin Server

Overview

An origin server is the server where your assets are stored. There are two types of origin servers, which are customer origin servers and CDN origin servers. A customer origin server is a web server that is external to our network, while a CDN origin server is a dedicated CDN storage server from which we allow customers to serve assets. You can take advantage of multiple origin servers to fulfill your CDN data delivery needs. This chapter will explain how you can configure each type of origin server.

Customer Origin Server

Before you can leverage assets stored or generated from a server external to the CDN network, you will need to configure a customer origin configuration on this platform. A customer origin configuration identifies one or more external servers and associates them with a unique folder name. When this folder name is specified as a part of the content access point (i.e., */80xxxx/Folder*), our servers will interpret it to identify the starting location for your assets (i.e., root folder of your origin server).

For example, if the primary purpose of your customer origin server is to serve advertising videos, then you might create a customer origin configuration whose **Directory Name** option is set to "ads." A sample CDN URL that can be used to access assets on this customer origin server is provided below.

- `rtmp://fms.xxxx.edgecastcdn.net/80xxxx/ads`

Note: In the above URL, you would replace `xxxx` with your CDN account number.

This CDN URL points to the root folder of your customer origin server (e.g., `www.server.com`). If you wanted to point to a particular folder on that server, you would simply append the desired path to the CDN URL. For example, the following sample CDN URL points to this location "`rtmp://www.myserver.com/marketing/2012/12`" on your customer origin server:

- `rtmp://fms.xxxx.edgecastcdn.net/80xxxx/ads/marketing/2012/12`

Tip: If you would like to use a more user-friendly URL, then you should take advantage of an edge CNAME. For more information, please refer to the **Masking the URL of a CDN Origin Server (CNAME)** section below.

Reminder: CDN and edge CNAME URLs are case-sensitive.

Hostnames/IP Addresses

Before our network can serve content from your server, a customer origin configuration that identifies that server must be created. A server can be identified within that customer origin configuration by either its hostname or IP addresses. Use one of the following formats to identify an origin server:

- `protocol://hostname:port` (e.g., `http://www.mydomain.com:80`)
- `protocol://IPv4Address:port` (e.g., `http://10.10.10.255:80`)
- `protocol://[IPv6Address]:port` (e.g., `http://[1:2:3:4:5:6:7:8]:80`)

Note: If a customer origin configuration points to a hostname, then it will be resolved to an IPv4 address in order to honor a client's request.

Note: Brackets are required when identifying an origin server through the use of IPv6 notation. This is the standard URI convention for IPv6 addresses.

You may use any combination of hostnames and IP addresses to identify the set of origin servers that can honor requests for the customer origin being configured. Defining more than one server per protocol will cause requests to be load balanced across those origin servers. For more information, please refer to the **Load Balancing** section below.

Important: An optimal configuration requires that the servers specified for each customer origin reside in relatively close vicinity to one another. If you would like to specify one or more servers that are located in a different geographic region, then we highly recommend that you create a separate customer origin configuration for those servers.

Note: A maximum of 10 hostnames and/or IP addresses can be associated with a customer origin configuration.

When specifying a hostname/IP address, you have the option to append a port through which communication will be established. If a port has not been specified, then a default port (i.e., 80) will be assigned to your hostname/IP address.

A sample CDN URL for HTTP requests for content stored on a customer origin server is provided below.

- `rtmp://fms.xxxx.edgecastcdn.net/80xxxx/CustomerOrigin/sample.flv`

The above sample CDN URL points to a customer origin configuration whose **Directory Name** option has been configured to "*CustomerOrigin*."

HTTP Host Header

HTTP 1.1 requires a Host header to be sent with each request. A Host header identifies the hostname/IP address and port associated with a request. This is especially useful when there are multiple virtual domains hosted on a single physical server or load-balanced set of servers.

Each customer origin configuration allows a Host header value to be configured. It is recommended that you set this value to either one of the hostnames/IP addresses from the HTTP hostnames lists, or else to an edge CNAME that references this customer origin server.

Load Balancing

The set of hostnames/IP addresses associated with a customer origin configuration are resolved into a list of IP addresses through DNS. If multiple unique IP addresses are associated with either option, then the selected load balancing mode determines which customer origin server will handle the next request.

The available load balancing options are:

- **Round Robin:** This mode will balance requests between all of the servers listed for a particular protocol. In other words, it will send the first request to the first server on the list. The next request will be sent to the next server on the list.
- **Primary & Failover:** This mode indicates that the specified servers form an ordered failover list. In other words, all requests will be forwarded to the first server on the list. If that server is unavailable, then the request will be sent to the next server on the list. This process continues until a server is able to honor the request. A server is unavailable when a TCP connection is refused or if the connection times out. The order of the failover sequence can be controlled by moving a hostname up or down through the up/down double arrow buttons.

Note: These load-balancing options are completely independent from any load balancing that may already exist on the customer origin server. For instance, traffic for a single IP address might be balanced across several physical servers.

Customer Origin Management

In order to take advantage of our CDN services, you will need to create a customer origin configuration for each customer origin server. This section explains how you can create, modify, and delete customer origin configurations.

Creating a Customer Origin Configuration

This section provides step-by-step instructions on how to create a customer origin configuration. Keep the following items in mind during this process:

- A customer origin configuration must point to a DNS record that has been fully propagated by your DNS provider. You may check this through the use of the "dig" command-line tool, which is a DNS query tool that allows you to query your DNS provider for the DNS record that corresponds to your hostname. When performing this dig, it is recommended that you use the "+trace" parameter. This will ensure that the DNS provider provides a direct response on the DNS record for the specified hostname. The syntax for this command is provided below.

```
dig hostname +trace
```

In the bottom section of the response, you should see an A record that points your hostname to an IP address. This indicates that your DNS record has been fully propagated. An excerpt from a sample response is shown below.

```
www.mycustomerorigin.com.    3600    IN      A       10.10.10.101
mycustomerorigin.com.       3600    IN      NS      dns02.dnsauthority.com.
mycustomerorigin.com.       3600    IN      NS      dns01.dnsauthority.com.
;; Received 100 bytes from 100.100.100.101#53(100.100.100.101) in 20 ms
```


- As a precautionary measure, it is recommended that you set the DNS TTL for your hostname to a low value until you have confirmed that our edge servers can properly communicate with your origin server. A low DNS TTL reduces the amount of time that an improper DNS configuration will affect your origin server, while increasing the number of DNS queries sent to your DNS provider.
- The required options for a new customer origin are a folder name, a hostname/IP address, and an HTTP host header. For your convenience, the HTTP host header is automatically populated when you add the first hostname/IP address.
- Make sure that your origin server does not restrict access to the IP address blocks listed on the **Customer Origin** page.
- It may take up to an hour for your new customer origin configuration to take effect.

To create a customer origin configuration

1. Navigate to the [Customer Origin](#) page which can be found on the **Flash** tab of the MCC.
2. In the **Directory Name** option, type the name of the folder that will be associated with the desired customer origin server. This folder will become a part of the contact access point in the CDN URL (e.g., `rtmp://fms.0001.edgecastcdn.net/800001/FolderName`).
3. Make sure that the **HTTP Edge Protocol** option is marked.
4. In the **Hostname or IP Address** option, type the base URL of the desired server (e.g., `http://flash.cdn.com:80`). Make sure to include the `http://` when specifying this URL. You should also include the port through which TCP communication can occur. Click **Add**. Repeat this step until you have finished adding all of the hostnames/IP addresses that will be associated with this customer origin.
5. In the **HTTP Host Header** option, specify either one of the hostnames from the HTTP hostnames lists or an edge CNAME that references this customer origin server.
6. Click **Add** to generate the customer origin configuration.
7. Make sure that all of the IP addresses listed on the **Customer Origin** page can access your server.

Note: It may take up to an hour before your new customer origin configuration takes effect.


Modifying a Customer Origin Configuration

A customer origin configuration can be modified at any time by clicking the  next to the desired customer origin. The configuration associated with that customer origin will appear. Simply make the desired changes and then click **Update** to apply them.

Note: If an edge CNAME points to a customer origin configuration, then you will not be allowed to modify the name of the folder associated with that customer origin configuration. If you would like to change the folder name, you will need to first delete the associated edge CNAME.

Note: It may take up to an hour for changes to your customer origin configuration to take effect.

Deleting a Customer Origin Configuration

A customer origin configuration can be deleted at any time by clicking the  next to the desired customer origin. Once you have confirmed the deletion, it will be removed from the list.

Note: If an edge CNAME points to a customer origin configuration, then you will not be allowed to delete the associated customer origin configuration. If you would like to delete it, you will need to first delete the associated edge CNAME.

Note: It may take up to an hour for changes to your customer origin configuration to take effect.

Content Expiration Headers

The web server hosting your content (i.e., customer origin server) can be configured to tell our edge servers how often to check for content freshness. If the requested content is stale and a newer version exists, then a new version of the requested content will be delivered to the POP where the request was received and it will replace the previously cached version of the asset.

The above configuration can be achieved by configuring your web server to set content expiration headers. For example, setting the header value "cache-control: max-age=86400" would cause our edge server to check for a new file every 86400 seconds (i.e., on a daily basis). If the revalidation with the origin server indicates that the requested asset is outdated or no longer exists, then the cached version of that asset will be replaced.

Note: If an edge server successfully revalidates a cached version of an asset, then the expiration time will automatically get updated. This means that the cached version of the asset will be assigned new Cache-Control and Expires headers based on the response returned from the origin server. If these headers have not been defined, then the asset will be assigned a default max-age of 7 days from the time it was successfully revalidated. Please refer to your web server's documentation for more information on how to configure these settings.

CDN Origin Servers

Our CDN provides storage servers on which you can store assets and then make them available through our CDN to your clients. These storage servers are called CDN origin servers. Content stored on CDN storage can be managed through your preferred FTP client. To find out the FTP URL, please refer to the **FTP** page of the **Media Manager** tab in the MCC. Your user name and password is the same as your MCC authentication.

Note: If the FTP page is not available, then you do not have sufficient permissions to log into the FTP server.

Note: You can also manage assets using the FTP client that has been embedded onto the **File Management** page of the **Media Manager** tab in the MCC. This FTP client does not require additional authentication to the CDN origin server.

Note: If you decide to use a third-party FTP client, then it is recommended that you use passive FTP (i.e., PASV).

Once you have uploaded the desired content to the CDN origin server, you can provide access to it through the following CDN URL:

- `rtmp://fms.xxxx.edgecastcdn.net/00xxxx`

Tip: You can use an edge CNAME to provide a more user-friendly URL to your clients.

Reminder: Make sure to replace xxxx with your CDN account number.

Masking the URL of a CDN Origin Server (CNAME)

The URL to the location on a CDN origin server where your Flash media content resides can be masked through the use of a Canonical Name (CNAME) record and an edge CNAME. An edge CNAME configuration informs our edge servers of the CNAME record that you would like to use, the domain used by that CNAME, and an optional path to a folder on your server.

Important: Informing our edge servers that you would like to use a CNAME record will not update or set that CNAME record on your DNS server. Before you can take advantage of the desired CNAME, you will need to set a CNAME record on your DNS server.

Example: Instead of streaming on-demand content using the following URL: `rtmp://fms.0001.edgecastcdn.net/000001/videos/technical`, you could create a CNAME record (e.g., `technical.videos.com`) to `fms.0001.edgecastcdn.net`. After which, you would create an edge CNAME that points the CNAME record to the "technical" folder. Content stored in that location can then be referenced using "`rtmp://technical.videos.com`" as the base URL in your Flash player.

Note: You may also take advantage of an edge CNAME to mask the URL of your customer origin server.

When configuring our edge servers to recognize a CNAME, you should keep in mind all of the following:


- Do not specify a protocol (i.e., `rtmp://` or `ftp://`).
- All alphabetical characters in the CNAME should be specified in lower-case letters.
- Set a CNAME record on your DNS server. The alias set on the DNS server should match the domain (e.g., `fms.xxxx.edgecastcdn.net`) associated with the edge CNAME configuration. If a directory path has been specified, then our servers will rewrite the URL to point to the appropriate folder.
- Adding, modifying, or deleting an edge CNAME may take up to an hour to take effect.

To configure our edge servers to recognize your CNAME


1. Navigate to the [Edge Cnames](#) page, which can be found on the **Flash** tab of the MCC.
2. In the **New Edge Cname** option, type the name of the desired CNAME record. The CNAME should be specified in lower-case letters and should not include the protocol (i.e., rtmp://).
3. Select whether the specified CNAME will point to a customer origin or CDN origin server.
4. By the **Points to** option, select the root location on the origin server to which the CNAME will be pointed. If you would like to indicate a specific folder, then you should type a forward slash (/) followed by the path to the desired folder.
5. Click **Add**.
6. Make sure that a CNAME record that points to the same domain has been registered on your DNS server. This CNAME record must match the name assigned to your edge CNAME.

Note: Keep in mind that your changes will take up to an hour to take effect.

To modify an edge CNAME

1. Navigate to the **Edge Cnames** page, which can be found on the **HTTP Large** tab of the MCC.
2. Click the  next to the edge CNAME that you would like to modify.
3. Make the desired changes.
4. Click **Update** to save your changes.
5. If you have modified the **New Edge Cname** option, make sure to update the CNAME record registered on the DNS server.

To delete an edge CNAME

1. Navigate to the **Edge Cnames** page, which can be found on the **HTTP Large** tab of the MCC.
2. Click the  next to the edge CNAME that you would like to delete.
3. When prompted, confirm the deletion of the selected edge CNAME.
4. It is recommended that you remove the CNAME record from the DNS server.

Securing Your Flash Media

Overview

You can secure the viewing of live and on-demand Flash media by time, country, URL, IP address, referrers, and SWF verification. Additionally, you can secure the publishing of your live streams through RTMPE and Live Authentication. We will discuss the many different ways in which your Flash media can be secured in this section.

Note: RTMP encryption and Live Authentication are only offered for the RTMP protocol. As a result, HTTP Progressive Download does not support these security measures. However, Token-Based Authentication can be used to protect all of your Flash content, regardless of protocol.

Preventing the Unauthorized Viewing of Your Flash Media

There are three different methods through which you can secure your Flash media against unauthorized viewing, which are tokens, SWF verification, and RTMPE. Each method is discussed below.

Note: Flash content made available through HTTP Progressive Download can only be secured using Token-Based Authentication.

Using Tokens to Secure Flash Media

Your Flash media can be protected according to the following criteria: expiration date, country, URL, IP address, and referrers. This information is stored as tokens in an encryption key. A media player will be denied access to Flash content secured in this manner when the appropriate token is not provided and the encrypted requirements of that token are not met.

Flash media can be secured by specifying the locations where Token-Based Authentication will be activated. This can be performed from the **Directories to Authenticate** section on the **Token Auth** page, which can be found on the **Flash** tab. Each specified location is secured recursively. In other words, all content that is published or stored in a location that starts with the specified path will be secured by Token-Based Authentication.

Note: For detailed information, please refer to the **Token-Based Authentication User Guide**, which is available from the Media Control Center (MCC).

Keep the following in mind when defining one or more directories that will be secured by Token-Based Authentication:

- The starting point for the relative path that will be secured by Token-Based Authentication is directly after the content access point (e.g., /000001).
- This relative path must start with a "/".
- If you secure the root folder (i.e., /), all live and on-demand streams will be secured by Token-Based Authentication.

Live StreamCast

For the purposes of determining whether a live stream should be protected by Token-Based Authentication, the origin path of the publishing point URL is used to determine whether a live stream will be secured. In the following example, your publishing point URL is the following:

```
rtmp://fso.lax.0001.edgecastcdn.net/200001/Videos/2012
```

Securing any of the following locations would require a token for a stream that is published to that location:

- /
- /Videos
- /Videos/2012

However, configuring Token-Based Authentication to secure any of the following locations would not secure your live stream (Presentation01):

- /Videos/2012/Presentation01
- /200001/

Important: Live streams cannot be protected on a per stream basis. Flash content can only be protected by securing a folder path.

Note: If you publish your live event to the root folder, then it will only be protected by Token-Based Authentication when the root folder is secured. Keep in mind that securing the root folder will require a token to be specified for all live and on-demand Flash content.

On-Demand Content

For the purposes of determining whether on-demand content should be protected by Token-Based Authentication, the path to the Flash media determines whether it will be secured. For example, if the URL for your on-demand content is:

```
rtmp://fms.0001.edgecastcdn.net/000001/Videos/2012/Presentation01.flv
```

Securing any of the following locations would require a token when attempting to stream on-demand content:

- /
- /Videos
- /Videos/2012

Reminder: Securing the root folder will secure all live and on-demand content through Token-Based Authentication.

However, configuring Token-Based Authentication to secure any of the following locations would not secure your on-demand content:

- /Videos/2012/Presentation01
- /000001/

Important: On-demand content cannot be protected on a per file basis. Flash content can only be protected by securing a folder path.

Accessing Secured Content

Secure content can only be accessed when a question mark and a valid token value is appended to the request for Flash content. Sample URL syntax for live and on-demand content is provided below.

- **Live:** `rtmp://fml.xxx.edgecastcdn.net/20xxx/Path/Stream?Token`
- **On-Demand:** `rtmp://fms.xxx.edgecastcdn.net/yyxxx/Path/Filename?Token`

Authenticating Your SWF File

You can configure our servers to limit connections to your streams to clients that use a predefined SWF file. Only clients that use the same build of that SWF file will be allowed to play content streamed through the Flash Media Streaming platform. This concept is known as SWF Verification.

Important: SWF Verification requires Adobe AIR or Flash Player 9.0.115.0 or higher.

Note: SWF file authentication is not supported for HTTP Progressive Download.

The purpose of this form of security is to force users to view your streaming content using your SWF file. Thus, it is most useful when you have created a custom SWF file that promotes your brand (e.g., logo, tag line, colors, etc.). This would prevent a malicious entity from streaming your content through a custom SWF file that hides your brand presence.

When setting up SWF Verification, you should keep the following in mind:

- The maximum number of SWF files that can be authenticated is limited to 50. However, you should keep the number of SWF files that will be authenticated to a minimum. This will ensure optimal performance.
- A SWF file may exist in any location that is accessible by our edge servers.
- In order to take advantage of SWF Verification, make sure that you do not use URL redirection when providing the Flash player to your clients. Our servers will only check for the Flash player at the given URL. If it detects a URL redirection, it will not authenticate the Flash player when your clients request it.
- If you plan to stream a live event using an encoder other than Flash Media Live Encoder, then you may need to contact your CDN account manager to authenticate that encoder to our ingest servers.
- It may take up to 15 minutes for changes to SWF Verification to take effect.
- A SWF configuration file (i.e., swfVerification.conf) is stored in a folder called "Ifms" on CDN storage. This folder and its contents should not be manually modified or deleted. Doing so may impair the functionality of the SWF Verification feature.

To enable SWF Verification

1. Mark the **Enable SWF Verification** option and click **Save**.
2. In the **New** option, type the URL to the desired SWF file.
3. Click **Add**.
4. Follow the **Check URLs** link to verify that the specified URL points to a SWF file. If the Verified column indicates that the SWF file could not be validated at that location, please double-check the URL and make the appropriate changes.

Encrypting Your Stream (RTMPE)

Flash Media Server offers the option to encrypt your Flash media stream through Real Time Messaging Protocol Encryption (RTMPE). RTMPE is a protocol offered by Adobe to encrypt your Flash stream and protect it from copyright infringements. RTMPE is the encrypted version of RTMP. In order for RTMPE to be most effective against copyright infringement, you should disable RTMP. This will ensure that only RTMPE can be used to stream your Flash media.

Note: Stream encryption is not supported for HTTP Progressive Download.

To protect your stream through RTMPE

1. Navigate to the **Advanced Settings** page, which can be found on the **Flash** tab of the MCC.
2. Click **Protocol Restriction** to display the [Flash Protocol Restriction](#) page.
3. Make sure that the **Enable RTMPE** option is marked.
4. Click **Save**.

Note: Keep in mind that your changes will take up to an hour to take effect.

Note: If you would like to provide maximum content protection, then you should also clear the **Enable RTMP** option.

Preventing the Unauthorized Publishing of Flash Media

Live Authentication prevents unauthorized users from publishing a live stream to your account by requiring that the encoder authenticate to the publishing server through a Live Authentication key. Each type of Live Authentication key is explained below.

- **Global Key:** Permits an encoder to publish to any location on the publishing server that has not been secured by a stream key.
- **Stream Key:** A stream key grants an encoder permission to publish the stream identified by the stream key. A stream key identifies a stream by its relative path and name. Both the relative path and the stream name must be specified in the **Stream Path** option and should use the following notation: *RelativePath/StreamName*. If you would like to permit a stream to be published to the root folder, then you may omit "*RelativePath/*" and only specify the name of the stream.
 - **RelativePath:** This path is relative to the publishing point URL (e.g., `rtmp://fso.lax.xxx.edgecastcdn.net/20xxxx`). It starts directly after the forward slash that follows the content access point (e.g., `20xxxx`). For example, if you would like to publish to

"rtmp://fso.lax.xxx.edgecastcdn.net/20xxx/Videos/2012," then the relative path for that stream would be "Videos/2012."

- **StreamName:** The stream name is the base name assigned to the stream within the encoder. If you are streaming using H.264 encoding, make sure to specify the filename extension (e.g., MyStream.mp4). By default, Flash Media Live Encoder uses MP4 for H.264 encoding.

Important: Live Authentication is a mandatory feature for Live StreamCast. It is provided for your protection and it prevents unauthorized users from hosting unauthorized streams through your account.

Reminder: A global key or a stream key that corresponds to the publishing point location must be specified as a part of the stream name (i.e., *StreamName?LiveAuthenticationKey*) when configuring an encoder's publishing point.

Sample Stream Key Configuration

The following example indicates the stream key settings and the encoder configuration required to push a stream called "MyStream" to the following publishing point location:

rtmp://fso.lax.xxx.edgecastcdn.net/20xxx/SampleContent/2012.

From the MCC, create a stream key with the following properties:

- **Stream Path:** SampleContent/2012/MyStream
- **Stream Key:** 123456789

From the desired encoder, encode a stream using the following publishing point:

- **Stream Name:** MyStream?123456789
- **Publishing Point URL:** rtmp://fso.lax.xxx.edgecastcdn.net/20xxx/SampleContent/2012

Live Authentication and Dynamic Streaming (Multi-Bit Rate Streams)

An encoder that has been configured to perform dynamic streaming will generate a stream for each configured bit rate. Each of those bit rate streams will need to be authenticated using either a global or a stream key. If you plan on using a global key, then it is simply a matter of specifying the global key for each specified stream key (e.g., *Stream%i?globalkey*). On the other hand, if you would like to authenticate using a stream key, then you will need to perform a few additional steps. These steps are outlined below.

To set up stream key support for a multi-bit rate stream

1. Decide upon a naming convention for each stream that will be broadcast by your encoder. For example, if you have configured your encoder to generate three bit rate streams, then you might decide to name them Stream01, Stream02, and Stream03.
2. Create a stream key with a unique stream path for each stream that will be published by your encoder.
3. Specify each stream name in your encoder. Make sure that this stream name does not include parameters (e.g., %i, %v, %a, etc.). You will also need to use a delimiter between each stream name/stream key combination. For example, Adobe Flash Media Live Encoder uses semi-colons to delimit stream names. Using the sample names from above, you would specify the following:
Stream01?streamkey01;Stream02?streamkey02;Stream03?streamkey03.

Note: For more information on how to publish a live stream, please refer to the **Publishing a Live Stream** section below.

Publishing a Live Stream

Before an encoder can publish your live stream, you will need to specify the publishing point and a name for the stream that you are publishing. The Live Authentication key must be specified along with the stream name. The proper syntax for a Live Authentication key is the following:

Syntax	Example
<i>StreamName?LiveAuthenticationKey</i>	WebCam01?Private

The term *LiveAuthenticationKey* stands for either a global or stream key. If you choose to use a stream key, make sure that the stream key corresponds to the specified publishing point location and name.

Note: The Live Authentication key is only used to authenticate that the stream being published is authorized. It should not be used when pointing your Flash player to your live stream. For information on how to secure your live streams, please refer to the **Preventing the Unauthorized Viewing of Your Flash Media** section above.

Appendix A

Migrating from Live Streaming to Live StreamCast

We highly encourage all customers that are currently using Live Streaming (Legacy) to migrate to Live StreamCast. This migration process consists of two items:

- Updating your encoder's configuration
- Updating your Flash player implementation

The changes required for both of these items are discussed below.

Updating your Encoder's Configuration

There are two settings that you will need to update on your encoder, which are the publishing point URL and the stream name.

Publishing Point URL

Live Streaming (Legacy) uses a URL similar to the following for the publishing point:

- `rtmp://flo.lax.xxxx.edgecastcdn.net/20xxx/<path>`

A sample publishing point URL for Live StreamCast is:

- `rtmp://fso.lax.xxxx.edgecastcdn.net/20xxx/<path>`

As you can tell, the only difference between the two URLs is the sub-domain. You will need to update the publishing point option on your encoder to use the FSO sub-domain.

Stream Name

There is a difference between how Live Streaming (Legacy) and Live StreamCast delimits a live authentication key from the stream name. Originally, Live Streaming (Legacy) used a forward slash (e.g., MyStream/LiveAuthenticationKey) to delimit the live authentication key from the stream name. With the introduction of Live StreamCast, Live Streaming (Legacy) can now use either a forward slash or a question mark. Live StreamCast, on the other hand, only supports a question mark delimiter (e.g., MyStream?LiveAuthenticationKey). Therefore, you will need to change your encoder's **Stream** option to use a question mark delimiter.

Important: Using a forward slash delimiter with Live Streaming (Legacy) will disable the Server-side archiving feature.

Updating your Flash Player Implementation

There are two steps required to migrate from Live Streaming (Legacy) to Live StreamCast. The first step involves updating your HTTP client's links to your stream. The second step is adding an FSubscribe call to your Flash player implementation.

Updating HTTP Client Links to your Stream

The player URL used by Live Streaming (Legacy) depends on your publishing point location. A sample player URL is provided below.

- `rtmp://fml.lax.xxx.edgecastcdn.net/20xxx/<path>/<stream>`

Live StreamCast only uses a single player URL for all publishing point locations. This player URL is provided below.

- `rtmp://fml.xxx.edgecastcdn.net/20xxx/<path>/<stream>`

As you can tell, the only difference between the two URLs is that a sub-domain that specifies a publishing point location has been removed. You will need to update the URLs used in your Flash player implementations to reflect this change.

Announcing your Stream to our Servers

Unlike Live Streaming (Legacy), Live StreamCast requires that you announce your live stream to our servers. This can be accomplished through an FSubscribe call. If this call is not properly included in your Flash player implementation, then your stream will not be viewable by that client. Sample code for JW Player and flowplayer is provided in the **Setting Up a Basic Live StreamCast** section of the **Live StreamCast** chapter.

Appendix B

Code Samples

We provide code samples to demonstrate how you can implement a Flash player that plays live or on-demand content from the Flash Media Streaming platform.

Live StreamCast

This section contains JW Player and flowplayer code samples for Live StreamCast.

JW Player Code Sample for Live StreamCast

Provided below is a code sample that demonstrates how to implement JW Player 5.3 and above with Live StreamCast. This sample code fragment is only provided to guide you through your configuration of your video player for use with our CDN.

Important: The provided code sample will not work with older versions of JW Player.

Note: If you have any questions on the implementation of your video player, please contact [LongTail Video](#).

Note: It is very important that you enable FSubscribe when implementing your video player. The line corresponding to the FSubscribe call is in bold font in the sample code fragment provided below.

When implementing the following code fragment, please keep in mind the following:

- Verify or update the path for the following Longtail Video resources:
 - `jwplayer.js`
 - `player.swf`
- Replace *StreamName* with the name of the stream as defined by your encoder.
- **Streamer:** Replace the value assigned to this variable with the publishing point URL. You can view the base URL from the Live StreamCast section of the [Flash Streaming](#) page of the **Flash** tab of the MCC. If you would like to manually type this URL, then you will need to do the following:
 - Replace `xxxx` with your CDN account number. This account number can be found on the upper-right hand corner of the MCC.

- If you are publishing to the root folder, then you should remove *"/Path"* from the URL. Otherwise, you should replace *Path* with the path to the location where the encoder is publishing the stream.

JW Player Code Sample

```
<head>
<script type='text/javascript' src='jwplayer.js'></script>
</head>
<body>
<div id="container">Loading the player...</div>
<script type="text/javascript">
  jwplayer("container").setup({
    flashplayer: "player.swf",
    height: 270,
    width: 480,
    file: "StreamName",
    provider: "rtmp",
    streamer: "rtmp://fml.xxx.edgecastcdn.net/20xxx/Path",
    'rtmp.subscribe': 'true'
  });
</script>
```

Note: The "rtmp.subscribe" variable instructs JW Player to make the FCSubscribe call to our servers. It is crucial that you include this line when implementing your video player.

Flowplayer Code Sample for Live StreamCast

Provided below is sample code that uses flowplayer (version 3.2.10) and the flowplayer streaming plugin RTMP (version 3.2.9) to stream live content through our CDN. This sample code fragment is only provided to guide you through your configuration of your video player for use with our CDN.

Note: If you have any questions on the implementation of your video player, please contact [flowplayer](#).

Note: The "subscribe" variable instructs flowplayer to make the FCSubscribe call to our servers. The "live" variable indicates that the content is a live stream. It is crucial that you include both of these lines when implementing your video player. The corresponding lines are indicated in bold font in the sample code fragment provided below.

When implementing the following code fragment, please keep in mind the following:

- Verify or update the path for the following flowplayer resources:
 - flowplayer-3.2.9.min.js
 - flowplayer-3.2.10.swf
 - flowplayer.rtmp-3.2.9.swf
- Replace *StreamName* with the name of the stream as defined by your encoder.
- Replace *xxxx* with your CDN account number. This account number can be found on the upper-right hand corner of the MCC.
- **netConnectionURL**: Replace the value assigned to this variable with the publishing point URL. You can view the base URL from the **Live StreamCast** section of the [Flash Streaming](#) page, which can be found on the **Flash** tab of the MCC. If you would like to manually type this URL, then you will need to replace *xxxx* with your CDN account number.
 - If you are publishing to the root folder, then you should remove *"/Path"* from the URL. Otherwise, you should replace *Path* with the path where the encoder is publishing the stream.

Flowplayer Code Sample

```
<head>
  <script src="flowplayer-3.2.9.min.js"></script>
</head>
<body>
  <!-- setup player normally -->
  <a class="player" id="LSC">
    
  </a>
  <script language="JavaScript">
    $f("LSC", "flowplayer-3.2.10.swf", {
      clip: {
        url: 'StreamName',
        // Configure clip to use RTMP as our provider.
        provider: 'rtmp',
        live: true
      },
      // Streaming plugins are configured under the plugins node
      plugins: {
        // RTMP plugin configuration.
```

```
rtmp: {
  url: 'flowplayer.rtmp-3.2.9.swf',
  // netConnectionUrl defines where the streams are found.
  netConnectionUrl: 'rtmp://fml.xxxx.edgecastcdn.net/20xxxx/Path',
  subscribe: true
}
});
</script>
</body>
```

On-Demand

This section contains JW Player and flowplayer code samples for On-Demand streaming.

JW Player Code Sample for On-Demand Streaming

Provided below is a code sample that shows how to implement JW Player 5.3 and above with on-demand content. This sample code fragment is only provided to guide you through your configuration of your video player for use with our CDN.

Note: If you have any questions on the implementation of your video player, please contact [LongTail Video](#).

When implementing the following code fragment, please keep in mind the following:

- Verify or update the path for the following Longtail Video resources:
 - jwplayer.js
 - player.swf
- Replace *Filename* with the filename of the Flash media file stored on the origin server.
- **Streamer:** Replace the value assigned to this variable with the URL to the folder where your Flash media is stored on the origin server. You can view the base URL for your customer origin servers and our CDN origin server from the **On-Demand Streaming** section of the [Flash Streaming](#) page, which can be found on the **Flash** tab of the MCC. If you would like to manually type this URL, then you should keep in mind the following:
 - Replace *yy* with the appropriate origin identifier. If you are pointing to a CDN origin server, then it should be set to 00. If you are pointing to a customer origin server, then it should be set to 80.
 - Replace *xxxx* with your CDN account number. This account number can be found on the upper-right hand corner of the MCC.

- Replace *Path* with the path to the Flash media file that will be streamed to your clients.

Note: If you are streaming a file from a CDN origin server, then the easiest way to find the correct URL for the streamer variable is to navigate to the [File Management](#) page, which can be found on the **Media Manager** tab of the MCC. From there, browse to the desired folder and then select the Flash media file that will be streamed. You should then select the **Flash** option and copy the URL that appears to the left of that option. Finally, you should replace the URL specified with the streamer variable with the URL copied from the **File Management** page.

JW Player Code Sample

```
<head>
  <script type='text/javascript' src='jwplayer.js'></script>
</head>
<body>
<div id="container">Loading the player...</div>
<script type="text/javascript">
  jwplayer("container").setup({
    flashplayer: "player.swf",
    height: 270,
    width: 480,
    file: "FiLename",
    provider: "rtmp",
    streamer: "rtmp://fms.xxxx.edgecastcdn.net/yyxxxx/Path"
  });
</script>
```

Flowplayer Code Sample for On-Demand Streaming

Provided below is sample code that uses flowplayer (version 3.2.10) and the flowplayer streaming plugin RTMP (version 3.2.9) to stream on-demand content through our CDN. This sample code fragment is only provided to guide you through your configuration of your video player for use with our CDN.

Note: If you have any questions on the implementation of your video player, please contact [flowplayer](#).

When implementing the following code fragment, please keep in mind the following:

- Verify or update the path for the following flowplayer resources:
 - flowplayer-3.2.9.min.js
 - flowplayer-3.2.10.swf
 - flowplayer.rtmp-3.2.9.swf
- Replace *Filename* with the filename of the Flash media file stored on the origin server. The notation varies according to file type.
 - **FLV:** Specify the filename using the following syntax: *flv:Filename* (e.g., *flv:Presentation01*). Make sure that you do not include the filename extension.
 - **H.264 F4V or MP4:** Specify the filename using the following syntax: *mp4:Filename.ext* (e.g., *mp4:Presentation01.mp4*). Make sure to include the filename extension.
- **netConnectionURL:** Replace the value assigned to this variable with the URL to the folder where your Flash media is stored on the origin server. You can view the base URL from your customer origin servers and our CDN origin server from the **On-Demand Streaming** section of the [Flash Streaming](#) page, which can be found on the **Flash** tab of the MCC. If you would like to manually type this URL, then you will need to do the following:
 - Replace *yy* with the appropriate origin identifier. If you are pointing to a CDN origin server, then it should be set to 00. If you are pointing to a customer origin server, then it should be set to 80.
 - Replace *xxxx* with your CDN account number. This account number can be found on the upper-right hand corner of the MCC.
 - Replace *Path* with the relative path to the Flash media file that will be streamed to your clients.

Note: If you are streaming content from a CDN origin server, then the easiest way to find this path is to navigate to the [File Management](#) page, which can be found on the **Media Manager** tab of the MCC. From there, browse to and then select the folder that contains the Flash media file that will be streamed. You should then select the **Flash** option and copy the URL that appears to the left of that option. Finally, you should replace the URL specified by the `netConnectionURL` variable with the URL copied from the **File Management** page.

Flowplayer Code Sample

```
<head>
  <script src="flowplayer-3.2.9.min.js"></script>
</head>
<body>
  <!-- setup player normally -->
  <a class="player" id="VOD">
    
  </a>
  <script language="JavaScript">
    $f("VOD", "flowplayer-3.2.10.swf", {
      clip: {
        url: 'Filename',
        // Configure clip to use RTMP as our provider.
        provider: 'rtmp'
      },
      // Streaming plugins are configured under the plugins node
      plugins: {
        // RTMP plugin configuration.
        rtmp: {
          url: 'flowplayer.rtmp-3.2.9.swf',
          // netConnectionUrl defines where the streams are found.
          netConnectionUrl: 'rtmp://fms.xxx.edgecastcdn.net/yyxxx/<path>'
        }
      }
    });
  </script>
</body>
```

Dynamic Streaming (Multi-Bit Rate)

This section explains how to set up dynamic streaming of live events and on-demand content through our CDN. The first step in setting up dynamic streaming is to choose a media player. The selected media player determines whether you should use a standard multi-bit rate configuration file or one that is specific to JW Player.

Once you have selected the desired media player, you will need to create a multi-bit rate configuration file that indicates all of the live events or on-demand content that will be streamed. The final step is to generate a media player that points to the configuration file.

Setting Up Dynamic Streaming for JW Player

This section explains how to:

- Generate a sample multi-bit rate configuration file.
- Generate an instance of the JW Player that points to the sample multi-bit rate configuration file.

Sample Multi-bit Rate Configuration File (JW Player 5.6 and above)

A multi-bit rate configuration file provides a listing of live streams or on-demand content that can be requested by a Flash player. The video that will be requested is determined by the bit rate parameter and the client's available CPU and bandwidth.

Provided below is sample multi-bit rate configuration file for JW Player 5.6 and above. This sample code is only provided to guide you through setting up multi-bit rate functionality for use with our CDN.

Note: If you have any questions on the implementation of your video player, please contact [LongTail Video](#).

When implementing the following sample code, please keep in mind the following:

- Make sure that all of the desired bit rate files are listed under the media:group tag. This may require that you modify or even create additional media:content entries in that tag. When modifying a media:content item, make sure to change the bit rate, stream/filename, and width (resolution) to values that are appropriate for the desired video file.
 - **url:** A live stream or the filename of the on-demand content must be specified in this parameter. The proper notation for each is provided below.
 - **Live Stream:** Specify the stream name (e.g., MyStream).

- **On-Demand Content:** Specify the filename and extension of each desired video (e.g., MyStream.flv).
- **Streamer:** The value that should be assigned to this parameter depends on whether you would like to use dynamic streaming with a live event or on-demand content.
 - **Live Event:** The streamer should be set to the publishing point URL. This URL, which is specified in the encoder, determines the location to which the live event will be pushed.
 - **On-Demand Content:** The streamer should be set to the folder where your Flash media is stored on the origin server. You can view the base URL for your customer origin server(s) and our CDN origin server from the **On-Demand Streaming** section of the [Flash Streaming](#) page, which can be found on the **Flash** tab of the MCC.

Multi-bit Rate Configuration File Sample

```
<rss version="2.0" xmlns:media="http://search.yahoo.com/mrss/"
xmlns:jwplayer="http://developer.longtailvideo.com/">
  <channel>
    <title>Playlist with RTMP Dynamic Streaming</title>
    <item>
      <title>Sample Title</title>
      <description>Sample Description</description>
      <media:group>
        <media:content bitrate="1200" url="Stream3.flv" width="640" />
        <media:content bitrate="800" url="Stream2.flv" width="480" />
        <media:content bitrate="350" url="Stream1.flv" width="240" />
      </media:group>
      <jwplayer:streamer>rtmp://fms.xxxx.edgecastcdn.net/yyxxxx/Path/</jwplayer:streamer>
    </item>
  </channel>
</rss>
```

More Information

After creating this configuration file, you will need to point your Flash player to it. Sample code for a JW player can be viewed from the **Dynamic Streaming Code Sample (JW Player 5.6 and above)** section below.

Dynamic Streaming Code Sample (JW Player 5.6 and above)

Before you can take advantage of multi-bit rate functionality for on-demand content, you will need to customize your Flash player to point to a multi-bit rate configuration file. Provided below is a code sample for JW Player 5.6 and above. This sample code fragment is only provided to guide you through your configuration of your video player for use with our CDN.

Note: If you have any questions on the implementation of your video player, please contact [LongTail Video](#).

When implementing the following code fragment, please keep in mind the following:

- Verify or update the path for the following Longtail Video resources:
 - swfobject.js
 - player.swf
- Replace *Playlist* with the full path to the multi-bit rate configuration file.

JW Player Code Sample

```
<head>
<script type='text/javascript' src='swfobject.js'></script>
</head>
<body>
<div id='mediaplayer'>This text will be replaced</div>
<script type='text/javascript'>
  var so = new SWFObject('player.swf','playerID','470','290','9');
  so.addParam('allowfullscreen','true');
  so.addParam('allowscriptaccess','always');
  so.addVariable('playlistfile','Playlist.xml');
  so.write('mediaplayer');
</script>
</body>
```

Setting Up Dynamic Streaming for SMIL Compliant Media Players

This section explains how to generate a sample multi-bit rate configuration file. For information on how to configure your media player to point to the configuration file, please refer to the documentation provided by your media player's manufacturer.

Sample Multi-bit Rate Configuration File (SMIL)

In the preceding sections, we examined how to leverage a multi-bit rate configuration file with JW Player. This section explains how to specify a multi-bit rate configuration file using a SMIL file instead of an XML playlist. A SMIL file uses similar notation as an XML file. Please refer to the documentation provided by your media player's manufacturer to discover whether it supports dynamic streaming using a SMIL file.

When implementing the following sample code, please keep in mind the following:

- **base:** The value that should be assigned to this parameter depends on whether you would like to use dynamic streaming with a live event or on-demand content.
 - **Live Event:** The streamer should be set to the publishing point URL. This URL, which is specified in the encoder, determines the location to which the live event will be pushed.
 - **On-Demand Content:** The streamer should be set to the folder where your Flash media is stored on the origin server. You can view the base URL for your customer origin server(s) and our CDN origin server from the **On-Demand Streaming** section of the [Flash Streaming](#) page, which can be found on the **Flash** tab of the MCC.
- Make sure that all of the desired bit rate files are listed under the switch tag. This will require that you modify or even create additional entries in that tag. When modifying an item in that tag, make sure to change the bit rate, stream/filename, and width (resolution) to values that are appropriate for the desired video file.
 - **src:** A live stream or the filename of the on-demand content must be specified in this parameter. The proper notation for each is provided below.
 - **Live Stream:** Specify the stream name (e.g., MyStream).
 - **On-Demand Content:** Specify the filename and extension of each desired video (e.g., MyStream.flv). When setting the filename of an H.264 file, you will need to prepend "mp4" (e.g., mp4:Video.f4v) to the name.

Multi-bit Rate Configuration File Sample

```
<smil>
<head>
<meta base="rtmp://fms.xxx.edgecastcdn.net/yyxxx/Path" />
  <layout>
    <root-layout width="320" height="240"/>
  </layout>
</head>
<body>
  <switch>
    <video src="mp4:Stream3.mp4" system-bitrate="500000"/>
    <video src="mp4:Stream2.mp4" system-bitrate="700000"/>
    <video src="mp4:Stream1.mp4" system-bitrate="1500000"/>
  </switch>
</body>
</smil>
```

More Information

After creating this configuration file, you will need to point your Flash player to it.

Glossary

A

Adaptive Streaming

This technology, which is based on HTTP progressive download, allows a player (e.g., Silverlight) to dynamically switch between different bit rate streams, in order to provide an optimal viewing experience based on a client's bandwidth and CPU usage. Smooth Streaming is an example of adaptive streaming.

Asset

This term refers to a resource that contains header information and a body that can be served to clients. Examples of assets include files and dynamic content.

C

Cache

This term refers to the storage of data to improve data delivery performance. When used in reference to our CDN, it refers to the temporary storage of an asset on an edge server or an origin shield server. Cache increases the speed through which that particular edge server can deliver that asset for subsequent requests.

CDN

Our content delivery network (CDN) consists of points-of-presence (POPs) that are placed at critical network and geographical locations around the world. This allows us to place content at the edge of the Internet allowing for faster downloads by your end-users.

CDN Domain

This term refers to a domain name assigned to your account. In the following examples of CDN domains, xxxx represents your CDN account number.

- `wac.xxxx.edgecastcdn.net`
- `wpc.xxxx.edgecastcdn.net`
- `fms.xxxx.edgecastcdn.net`
- `wms.xxxx.edgecastcdn.net`

CDN Origin

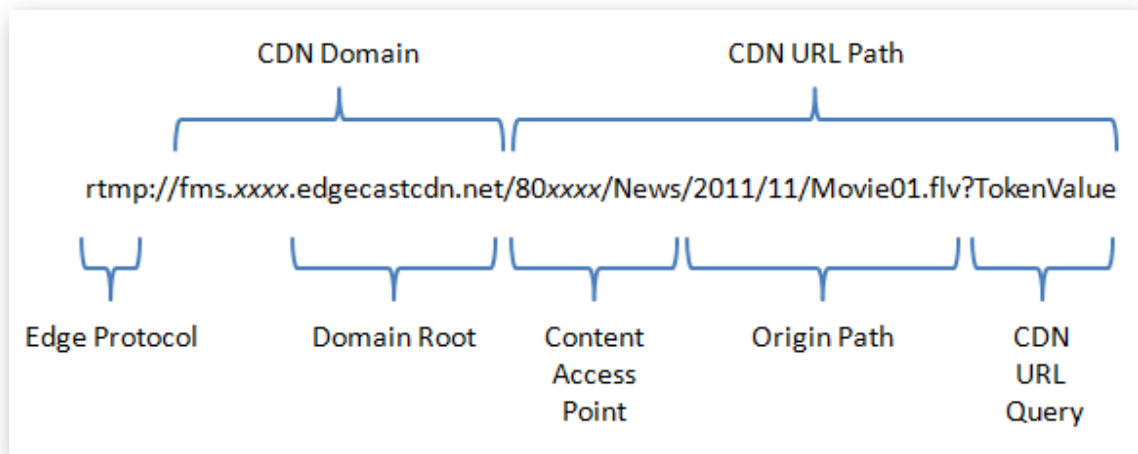
This term refers to a storage server on our CDN. Our CDN origin servers are in close proximity to our POPs, in order to provide optimal conditions for transferring data from a CDN origin server to your end-users via our POPs.

CDN Origin Identifier

This type of identifier in the CDN URL indicates that requested asset should be retrieved from the CDN origin server. A CDN origin identifier is indicated by "00" as the starting two numbers in the CDN URL path.

CDN URL

This type of URL identifies a location or an asset on our content delivery network. The following diagram indicates the different components in a CDN URL. Keep in mind that xxxx represents your CDN account number.



CDN URL Path

This term refers to the portion of the CDN URL that appears after the CDN domain. It provides the relative path to a folder or an asset on either a CDN or customer origin server. In the following examples of CDN URL paths, xxxx represents your CDN account number.

- `/00xxxx`
- `/00xxxx/Videos/2012/11/`
- `/00xxxx/Videos/2012/11/Presentation01.flv`

CDN/Edge CNAME URL Query

This term refers to the query string that appears after a question mark in a CDN or edge CNAME URL. If Token-Based Authentication has been enabled on this platform, then the query string should either be blank or a token value should appear directly after the question mark.

CNAME

A Canonical Name (CNAME) record is used to indicate that a domain name is an alias of another domain name. A CNAME record must be registered on a Domain Name System (DNS). This term should not be confused with edge CNAME.

Content Access Point

It provides a point of reference to any folder on a CDN or customer origin server. This relative path starts directly after the CDN domain. The proper syntax for a content access point is `"/yyxxx/path,"` where *yy* stands for the identifier and *xxx* stands for the CDN account number. The term *path* is optional and stands for the path to the folder specified by an edge CNAME configuration.

Customer Origin

This term refers to a storage server that is external to our CDN. Assets can be delivered from your storage server to your end-users via our POPs.

Customer Origin Identifier

This type of identifier in the CDN URL indicates that requested asset should be retrieved from the customer origin server. A customer origin identifier is indicated by "80" as the starting two numbers in the CDN URL path.

D

Domain Root

This term identifies the top and second-level domains associated with the CDN domain name. An example of a domain root is "google.com."

E

Edge CNAME

This term refers to the mapping of a CNAME record to a directory on a CDN or customer origin server. The purpose of this mapping, which is only used by our CDN, is to establish a user-friendly alias for content served through the CDN. It relies upon your CNAME record being properly mapped on a DNS server.

Edge CNAME URL

This type of URL takes advantage of an edge CNAME to mask a CDN URL. This allows it to identify a location or an asset on our content delivery network using a more user-friendly URL. An edge CNAME URL is specific to the platform (i.e., HTTP Large Object, HTTP Small Object, or Flash Media Streaming) from which it was configured.

In the following examples, the domain assigned to the edge CNAME is "www.mydomain.com."
 In the first example, the edge CNAME references the following CDN URL:
 "rtmp://fms.xxxx.edgecastcdn.net/00xxxx." In the following two examples, the edge CNAME
 references the following CDN URL: "rtmp://fms.xxxx.edgecastcdn.net/00xxxx/Videos."

Edge CNAME URL	Points To
rtmp://www.MyDomain.com/	rtmp://fms.xxxx.edgecastcdn.net/00xxxx/
rtmp://www.MyDomain.com/2012/11/	rtmp://fms.xxxx.edgecastcdn.net/00xxxx/Videos/ 2012/11/
rtmp://www.MyDomain.com/2012/11/Presentation01.flv	rtmp://fms.xxxx.edgecastcdn.net/00xxxx/Videos/ 2012/11/Presentation01.flv

Edge CNAME URL Path

This term refers to the portion of the edge CNAME URL that appears after the edge CNAME. It provides the relative path to a folder or an asset on a CDN or customer origin server. In the following examples of edge CNAME URL paths, the edge CNAME points to the following CDN URL: "rtmp://fms.xxxx.edgecastcdn.net/00xxxx/Videos."

Edge CNAME URL Path	Actual Edge CNAME URL
/2012/11/	rtmp://fms.xxxx.edgecastcdn.net/00xxxx/Videos/2012/11/
/2012/11/Show01.flv	rtmp://fms.xxxx.edgecastcdn.net/00xxxx/Videos/2012/11/Show01.flv

Edge Protocol

This term refers to the protocol (e.g., HTTP, RTMP, and MMS) used in a CDN URL or an edge CNAME URL.

Edge Server

This type of server is located near the edge of the Internet where its close proximity to your end-users allows it to deliver data more quickly than normal Internet communications. Our edge servers are integral component of our POPs.

Encryption Key

Token-Based Authentication requires the use of an encryption key to encrypt and decrypt token values. There are two types of encryption keys, which are a primary and a backup key. Both of these keys can be used to encrypt and decrypt token values.

F

Flash Live StreamCast

Please see Live StreamCast.

Flash On-Demand

This term refers to the streaming of Flash media content stored on an origin server through our CDN. This type of streaming uses Real-Time Messaging Protocol (RTMP, RTMPE, RTMPT, or RTMPTE) to deliver video to your clients.

G

Global Key

This type of Live Authentication key can be used to authenticate all live Flash streams. Only a single global key can be specified.

H

HTTP Large Object

This platform consists of dedicated edge servers that retrieve, cache, and serve large assets to your clients. These servers have been optimized to cache assets. A typical asset for the HTTP Large Object platform is larger than 300 KB.

HTTP Progressive Download

This method of streaming video content is performed through the HTTP protocol. Progressive downloads are not as secure as other streaming methods, since the entire asset will be stored on your client's computer. This allows your client to save and share your content with other users.

HTTP Small Object

This platform consists of dedicated edge servers that retrieve, cache, and serve smaller content to your clients. These servers have been optimized to index files. A typical asset for the HTTP Small Object platform is smaller than 300 KB.

I

Identifier

It identifies how a request will be routed through our CDN. Examples of identifiers are:

- **00:** CDN origin identifier
- **80:** Customer origin identifier
- **20:** Flash Media Streaming (Live StreamCast)

Ingest

This term refers to the process of capturing and transforming video into a stream.

Ingest Server

This term refers to the type of server that is dedicated to the process of capturing and transforming video into a stream. This type of server will then broadcast that stream throughout our CDN.

L

Live Authentication Key

This type of key authenticates a stream on the Flash Media Streaming platform before it is ingested by our publishing server. There are two types of live authentication keys, which are global and stream keys. When configuring your encoder, you must specify the stream name, a token delimiter, and then a global or stream key: `<StreamName>?<LiveAuthenticationKey>` (e.g., `MyStream?1234567890`).

Live Ingestion Point

This term refers to the location on a server where our CDN can access encoded media. There are two types of live ingestion points, which are pull source and publishing point.

Live StreamCast

This term refers to the streaming of a live Flash media stream through our CDN. This type of streaming uses Real-Time Messaging Protocol (RTMP, RTMPE, RTMPT, or RTMPTE) to deliver video to your clients.

Live Streaming (Legacy)

This term refers to the legacy delivery method for streaming Flash media content. We strongly recommend that you take advantage of Live StreamCast to take care of your Flash streaming needs.

Live Streaming Identifier

This type of identifier in the CDN URL indicates that requested asset should be streamed from the live ingestion point. A live stream identifier is indicated by "20" as the starting two numbers in the CDN URL path.

Load

This feature allows you to cache an asset on all of our POPs. This feature is unsupported for use with the Windows Media Streaming platform or Live StreamCast (Flash Media Streaming).

M

Media Control Center (MCC)

This web application is provided to help you manage all of your CDN needs. The major features that are available from the MCC are CDN configuration settings, cache management, file management, reports, and analytics. Additionally, the MCC allows you to configure your organization's settings, such as granting or denying access to the MCC. You can access the MCC through the following URL:

<https://my.edgecast.com>

O

Origin Path

It references a relative path to a folder or an asset in a CDN URL. This type of path follows the content access point.

Origin Server

This term refers to the servers that store the assets that will be distributed by our POPs. There are two types of origin servers, which are CDN origin and customer origin servers.

Origin Shield

This feature provides a layer of protection for your customer origin server by creating an intermediate caching layer between it and our edge servers. This caching layer resides on one or more of our point-of-presence (POPs). Requests that have not been previously cached on a POP will be channeled through the closest origin shield server. The origin shield server will then either serve a cached version of the requested content or retrieve it from your customer origin server. This feature reduces the amount of bandwidth used on your customer origin server, since most requests will be handled by the origin shield server.

P

Player URL

A media player uses this type of URL to stream content. It identifies the location of the streamer on the live ingestion point.

Point-of-Presence (POP)

A point-of-presence, or data center, is an access point to the Internet. The main components of a POP are edge servers, CDN origin servers, and publishing servers.

Pre-Cached

A pre-cached asset means that it has been loaded to all of our POPs. Pre-caching your assets allows even quicker content delivery to your clients, since it ensures that the requested asset will not have to be retrieved from the origin server.

Publishing Point

This term refers to the location on the publishing server to which your encoder will broadcast encoded media.

Publishing Server

This term refers to a CDN server that will redistribute encoded media as a streamer that will be broadcast to your end-users via our POPs.

Pull Source

This term refers to the location on an external server from which a broadcasted stream will be retrieved by our publishing server.

Pull Stream

This type of stream requires that our servers retrieve, or pull, a live stream from a server with a public IP address.

Purge

This feature allows you to remove the cached version of an asset from all of our edge servers and origin shield servers. A purge can be performed on a folder or an individual asset.

Push Stream

This type of stream requires your encoder to send, or push, encoded video to a CDN server. From there, our server will create a stream and deliver it to clients that request it.

Q

Query String

Additional data can be appended to a URL (e.g., <http://www.server.com/index.html?Data=xyz>). If Token-Based Authentication has been enabled on this platform, then the query string should either be blank or it should contain a token value directly after the question mark. Query string information can be stored in our log files. Keep in mind that query string caching is not supported on the Flash Media Streaming or the Windows Media Streaming platforms.

R

Request

A request consists of a set of headers and a body sent from a client. This header data and the body define the requested content. Typically, a request is sent from a client to an edge server. If the requested content is not found, then our edge servers will forward this request to an origin server.

Response

A response consists of the headers and the body sent from a server responding to a request. If an origin server is returning a response, then this response will be sent to an edge server. The edge server will then forward the response to a client.

S

Server Side Archiving

This feature allows you to archive live Flash streams on an origin server. This allows you to provide video on-demand capabilities to live Flash streams.

Stream

A stream consists of the delivery of audio/video content in a format that allows your clients to play it back through a multimedia player.

Stream Key

This type of Live Authentication key can only authenticate a Flash stream when it is published to the path associated with it.

T

Time to Live (TTL)

This term refers to the amount of time that a cached asset is still considered fresh. Our edge servers will continue to serve a cached version of an asset while its TTL has not expired. An asset's TTL is calculated by the Cache-Control and Expires headers associated with the response sent by a CDN or customer origin server.

Token

A token or a token value must be provided when a client requests content protected with Token-Based Authentication. Each token value contains security requirements that have been encoded using an encryption key. A token value can be specified by appending a question mark and the token value to the CDN URL path.

Token-Based Authentication

It requires a token value to be supplied when a user requests an asset from a protected folder. This token value is then decrypted on our server. If the user meets the specified requirement(s), then the asset will be delivered. Otherwise, the user will be denied access to the asset.

W

Windows Media Live Streaming

This term refers to the streaming of a live Windows media stream through our CDN. Although this type of streaming can use MMS as the protocol identifier, it will actually use either the HTTP or RTSP protocol to deliver your video to your clients.

Windows Media On-Demand Streaming

This term refers to the streaming of Windows media content stored on our CDN storage service. Although this type of streaming can use MMS as the protocol identifier, it will actually use either the HTTP or RTSP protocol to deliver your video to your clients.