EdgeCast Networks, Inc.

# Token-Based Authentication 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.

FLASH is a registered trademark of Adobe Systems Incorporated.

WINDOWS is a registered trademark of Microsoft Corporation.

## About This Guide

Token-Based Authentication Administration Guide
Version 2.11
5/2/2012

# Table of Contents

Introduction ........ 1

Overview ........ 1

How Does It Work? ........ 1

Choosing a Platform to Secure ........ 2

Setting Up Token-Based Authentication ........ 4

Overview ........ 4

Setting an Encryption Key ........ 4

Changing Your Encryption Key ........ 5

Protecting Your Content by Folder ........ 6

HTTP-Based Platforms (Sample Scenarios) ........ 8

Flash Media Streaming Platform ........ 9

Windows Media Streaming Platform ........ 11

Authentication Folder Administration ........ 13

Protecting your Content by Request Type ........ 14

Interaction with CDN Settings ........ 14

Determining How to Protect Your Content ........ 15

Overview ........ 15

Setting Content Expiration Date ........ 15

Allowing or Blocking Users by Country ........ 16

Preventing the Reuse of Token Values ........ 16

Allowing or Blocking Users by Host ........ 20

Allowing or Blocking Users by Referrer ........ 21

Allowing Users by IP address ........ 24

Allowing or Blocking Users by Protocol ........ 24

Preventing Changes to Bandwidth Throttling Settings ........ 25

Generating Tokens ........ 26

Overview ........ 26

Manually Generating a Token ........ 26

Using Our Token Generation Application ........ 27

Building a Token Generator ........ 28

Decrypting an Existing Token ........ 28

Providing Access to Protected Content ........ 29

Overview ........ 29

HTTP Large Object Example ........ 29

HTTP Small Object Example ........ 29

ADN Example ........ 30

Flash Media Streaming Example ........ 30

Windows Media Streaming Example ........ 31

Redirecting Unauthorized Users ........ 31

Quick Reference ........ 33

Security Parameters ........ 33

Bandwidth Throttling Reference ........ 36

Appendix A ........ 37

Country Codes (ISO 3166) ........ 37

Appendix B ........ 46

Flash Content Security Scenarios ........ 46

Flash Media Streaming (Live StreamCast) ........ 46

Flash Media Streaming (On-Demand Content) ........ 47

Glossary ........ 49

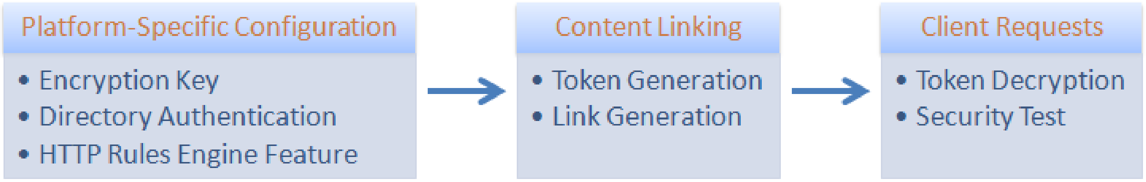
# Introduction

## Overview

Token-Based Authentication provides security for assets accessed through our content delivery network. For example, you can secure your content by the country, URL, IP address, protocol, or the referrer used to request access to your content. Additionally, you can protect your content by only allowing it to be available for a certain amount of time. Regardless of how you decide to protect your content, only authorized users that provide a valid token when requesting an asset will be able to access your content.

## How Does It Work?

There are three main Token-Based Authentication phases, which are configuration, content linking, and client requests. The order under which these phases should take place is illustrated below.



***Token-Based Authentication Setup Overview***

### Platform-Specific Configuration

Before content can be secured, you will need to determine which platforms will be protected by Token-Based Authentication and then configure each desired platform. Token-Based Authentication configuration consists of specifying at least one encryption key and then determining how content will be secured.

One way in which content can be secured is by specifying a directory. All requests for content in that directory or a sub-folder of that location will be secured. An additional method for securing content is only available if you have purchased HTTP Rules Engine. A rule can be created within HTTP Rules Engine that enables or disables Token-Based Authentication when a request meets predefined criteria. For example, you can specify that all Microsoft Word documents will be protected by Token-Based Authentication.

### Content Linking

Once Token-Based Authentication has been configured on the desired platform, encrypted tokens can be generated that define which users will be able to access your content. Protected content can then be made available to your clients by providing a link that includes a CDN or edge CNAME URL to the desired asset and a query string parameter that contains the token value that defines the security parameters that must be met before a user can access the requested content.

**Note:** Assets that reside in unprotected folders cannot be secured through the use of security parameters. An unsecured asset may be accessed using a standard CDN or edge CNAME URL.

### Handling Client Requests for Secured Content

When a client attempts to access an asset protected by Token-Based Authentication, they will need to provide a properly formatted URL and meet the security requirements defined in the HTTP request. A more detailed explanation is provided below.

1. An authorized request must contain a valid security token that is appended to the file name in the CDN or edge CNAME URL (e.g., http://data.server.com/asset.txt?c1019f8a6942b46a1ce679e168d5797670f3ee7e39068054ee4534d8a5a859dc06).
2. Our edge servers will decrypt the token using either the current primary or backup encryption key for the platform associated with the request. The decrypted value will reveal the security requirements for the requested content.
3. The user must satisfy all of the security requirements defined for the requested content. If the user meets the requirements defined in the decrypted token value, then access will be granted to the requested content. Otherwise, the user will be denied access. If a user is denied access, then you may choose to redirect the user to another web page.

## Choosing a Platform to Secure

Token-Based Authentication can secure content stored on any of our platforms (e.g., HTTP Large Object, HTTP Small Object, Flash Media Streaming, etc.). However, you will need to configure each platform individually. This provides complete control over how content is secured on each platform. The following table lists the available features and security parameters for each platform.

**Note:** The term "HTTP-Based Platform" encompasses the HTTP Large Object, HTTP Small Object, and the Application Delivery Network (ADN) platforms.

| Feature/Security Parameter | HTTP-Based Platform | Windows Media Streaming | Flash Media Streaming |
|---|---|---|---|
| Primary Key | ● | ● | ● |
| Backup Key | ● |  | ● |
| Folder-Level Security | ● | ● | ● |
| Expiration Date Parameter (ec_expire) | ● | ● | ● |
| Allow URL Parameter (ec_url_allow) | ● |  | ● |
| Allow Country Parameter (ec_country_allow) | ● | ● | ● |
| Deny Country Parameter (ec_country_deny) | ● | ● | ● |
| Allow Host Parameter (ec_host_allow) | ● |  |  |
| Deny Host Parameter (ec_host_deny) | ● |  |  |
| Allow Referrer Parameter (ec_ref_allow) | ● | ● | ● |
| Deny Referrer Parameter (ec_ref_deny) | ● | ● | ● |
| Allow Protocol Parameter (ec_proto_allow) | ● |  |  |
| Deny Protocol Parameter (ec_proto_deny) | ● |  |  |
| Allow Client IP Address Parameter (ec_clientip) | ● | ● | ● |
| Token Encryption/Decryption | ● | ● | ● |
| Custom Denial Handling | ● |  |  |

**Note:** HTTP-based platforms support the use of the ec_host_allow and the ec_host_deny parameters, even though they are not available from the **Encrypt Tool** section of the **Token Auth** page. For more information, please refer to the **Allowing or Blocking Users by Host** section in the **Determining How to Protect Your Content** chapter.

**Note:** Although the Windows Media Streaming platform supports the use of the ec_ref_allow and the ec_ref_deny parameters, these parameters have a slightly different behavior than when used with other platforms and they are not available from the **Encrypt Tool** section of the **Token Auth** page. For more information, please refer to the **Allowing or Blocking Users by Referrer** section in the **Determining How to Protect Your Content** chapter.

# Setting Up Token-Based Authentication

## Overview

There are two main aspects for setting up Token-Based Authentication, which are setting an encryption key and determining the location(s) that will be protected by it. Both of these items must be configured on each platform that will be secured by Token-Based Authentication. If either setting is not properly configured, then content on that platform will not be properly secured. This chapter explains how you can properly set up Token-Based Authentication on a per platform basis.

## Setting an Encryption Key

Content on a particular platform will not be protected by Token-Based Authentication until an encryption key has been assigned to it. Once a key has been assigned to a particular platform, all requests for assets in protected folders for that platform will require a token value. When an edge server receives a request for a secure asset, it will decrypt the associated token value using the encryption key configured for that platform.

An encryption key can consist of any combination of alphanumeric characters. All other characters, including spaces, are not valid for encryption keys. You should also keep in mind that encryption keys are case-sensitive. In other words, the case of an encryption key determines how token values will be decrypted.

Setting an encryption key is as simple as navigating to the **Token Auth** page for the desired platform in the MCC, assigning a value to the **Primary Key** option, and then clicking **Update**. Once you have set a primary key, you will need to specify which folders you would like to protect.

When assigning an encryption key for the first time, keep in mind that your changes (i.e., setting an encryption key and specifying the folder(s) to be protected) may take up to an hour to take effect. During this time period, the content of your protected folders can be accessed normally.

**Important:** By default, a token value is only specific to an encryption key and not to a folder or a platform.  This means that it is possible for a client to use a single token value to gain access to protected content from various folders across different platforms. If you would like to ensure that token values cannot be reused on different platforms, then you should define a unique encryption key on each platform.

# Changing Your Encryption Key

The encryption key assigned to a platform is crucial for decrypting token values. If the encryption key used to generate a token value is no longer set for that platform, then access will be denied to that asset. The following factors may prevent you from instantly switching to a new encryption key:

- The amount of time it takes to update all of your links to protected content.
- Cached assets that contain links to protected content using old token values.
- The amount of time it takes for your new encryption key to take effect (approximately 1 hour).

As a result of all of these factors, it is recommended that you have two active encryption keys to ensure that authorized users enjoy uninterrupted access to your assets. This procedure would require that you assign your old key as a backup key when creating a new encryption key. Since your old key is still an active encryption key, users will still be able to access your data using old tokens. You can remove your encryption key after the new encryption key has taken effect, all of your links have been updated, and your old assets are no longer being served. This process will ensure a smooth transition to a new encryption key.

**Note:** Windows Media Streaming does not have a backup key. If you have to change your encryption key, it is recommended that you do so at a time when demand for your videos is at its lowest.

**To change your encryption key (recommended procedure)**

1. From the MCC, navigate to the **Token Auth** page for the desired platform.
2. From the **Token-Based Authentication** section, copy the value from the **Primary Key** option to the **Backup Key** option.
3. In the **Primary Key** option, type your new encryption key.
4. Click **Update** to save your changes. It may take up to an hour for your primary key to become active.
5. After an hour has elapsed, update all of your links that point to protected content to use tokens generated with the new encryption key.
6. Once all of your links have been updated, purge all of the assets that point to protected content. Purges can be performed from the **My Edge** page, which is available for each of the platforms, in the MCC. Keep in mind that you may purge a folder recursively.
7. Clear the **Backup Key** option. Click **Update** to save your changes. It may take up to an hour for your backup key to become deactivated. After which, links that use token values based on the old encryption key will be rejected.

# Protecting Your Content by Folder

Before discussing how to protect your content by folder, let us review the following points:

- Token-Based Authentication secures content by platform (e.g., HTTP Large Object, HTTP Small Object, Flash Media Streaming, etc.).
- A primary and/or backup encryption key must be specified for each desired platform.
- An encryption key by itself will not secure your content on that platform.

The content that you would like to secure through Token-Based Authentication needs to be identified. One way of accomplishing this is to specify the location(s) that will be secured by Token-Based Authentication on a per platform basis. The **Directories to Authenticate** section allows you to define one or more protected locations using a relative path to the desired folder. The starting point for this relative path is defined below:

| URL Type | Relative Path (Starting Point) |
|---|---|
| CDN URL | Specify a relative path that starts directly after the content access point (e.g., /000001, /200001, or /800001).<br>In the sample URL below, the gray text indicates what should be excluded when securing a location. This sample request would be secured by any of the following configurations: /, /customerorigin, and /customerorigin/videos.<br>http://wac.0001.edgecastcn.net/800001**/customerorigin/Videos/**fly.flv |
| Edge CNAME URL (CDN Origin) | Specify a relative path that starts directly after the hostname.<br>In the sample URL below, the gray text indicates what should be excluded when securing a location.<br>http://www.domain.com**/presentations/**businessplan.ppt |
| Edge CNAME URL (Customer Origin) | Specify a relative path that starts with the name of the customer origin configuration referenced by the edge CNAME URL.<br>Two sample URLs are provided below. The first URL is the edge CNAME URL that is requested by a client. The second URL is the CDN URL generated by our servers from the edge CNAME URL. The CDN URL is used to determine the starting point for the relative path that should be secured.<br>In the sample CDN URL below, the gray text indicates what should be excluded when securing a location.<br>This sample edge CNAME URL:<br>http://www.domain.com/Videos/fly.flv<br>is translated to the following CDN URL:<br>http://wac.0001.edgecastcn.net/800001**/customerorigin/Videos/**fly.flv |

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

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

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

### Securing Content on a Customer Origin Server

If you would like to protect a location other than the root folder on a customer origin server, then you must include the name assigned to the customer origin configuration (e.g., /MyCustomerOrigin) when specifying the location that will be authenticated. This configuration is required regardless of whether you plan on using a CDN or edge CNAME URL to access your content. Failure to include the customer origin name may prevent that location on your customer origin server from being protected by Token-Based Authentication.

**Important:** Although an edge CNAME URL does not include the name of a customer origin server and may not include the path to the desired folder, it will be treated as if the corresponding CDN URL had been used. As a result, when securing such a location you must specify the name of the customer origin server followed by the relative path to the desired folder (e.g., /MyCustomerOrigin/Presentations/2012).

**Note:** There is an exception that only applies to the HTTP Large Object, HTTP Small Object, and the ADN platforms. A customer origin configuration name does not have to be specified when it contains a period (e.g., www.domain.com). However, for the purpose of clarity and consistency, it is still recommended to do so.

### Scope

When choosing which folders to protect, keep in mind that security is applied recursively to that folder. This means that all assets residing in the specified folder or its subfolders will be protected by Token-Based Authentication.

**Important:** Protecting a folder's content through Token-Based Authentication will only provide security through the configured platform. This means that a user could potentially download the secured content by using a URL for a different platform (e.g., HTTP Large Object instead of HTTP Small Object). If you would like to ensure that your content is protected across all platforms, then you should configure Token-Based Authentication on the desired folders across all of your platforms.

If you do not wish to protect your content by specific folders, you can choose to protect all of your content for a specific platform. This can be accomplished by simply making sure that the **New** option, which can be found in the **Directories to Authenticate** section, is set to forward slash (/). Click **Add**. This procedure will add the root folder to the list of protected folders. Due to the recursive nature of Token-Based Authentication, the contents of the root folder and all of its subfolders will be secured. Therefore, you do not need to specify any additional folders under the **Directories to Authenticate** section for this platform.

**Note:** Protecting the root folder (/) will also secure content stored on each customer origin server.

## HTTP-Based Platforms (Sample Scenarios)

This section illustrates how the following URLs interact with Token-Based Authentication:

1. http://wpc.0001.edgecastcdn.net/000001/Secure/index.html
2. http://wpc.0001.edgecastcdn.net/000001/Secure/Data/index.html?c1019f8a6942b46a1ce679e66cd579767
3. http://wpc. 0001.edgecastcdn.net/800001/MyServer/Secure/index.html
4. http://secure.server.com/index.html?c1019f8a6942b46a1ce679e66cd579767

**Note:** Although the above sample URLs are specific to the HTTP Large Object platform, the analysis provided below also applies to the HTTP Small Object and the ADN platforms.

We will now examine the effect of securing a location called "/Secure" will have on the above URLs.

1. The first URL points to an asset stored in a folder called "Secure" on a CDN origin server. Since the asset is stored in a folder protected by Token-Based Authentication, it requires a token. Since a token was not specified for this request, the asset will not be served to the client.
2. The second URL points to an asset stored on a CDN origin server. Since this asset is located in a subfolder of a protected folder, it is also protected by Token-Based Authentication. The requested asset will be delivered to the client, as long as the token is valid and the user requesting it meets the requirements specified in the provided token.
3. The third URL points to a customer origin server. The "MyServer" folder is the name assigned to the customer origin configuration for the server hosting your assets. The requested asset is unprotected, since its relative path starts with "/MyServer" instead of with "/Secure." As a result, it will be served to the client.
4. The fourth URL is an edge CNAME URL. In this particular case, this edge CNAME takes advantage of a customer origin configuration called "MyServer" and points to a folder called "Secure." Although the edge CNAME URL points to the "Secure" folder, the relative path for this type of URL (i.e., edge CNAME URL that points to a customer origin server) starts with the customer origin name (i.e., /MyServer). As a result, the unprotected asset will be served to the client.

We have just examined how several URLs would be affected when the "/Secure" location was secured on an HTTP-based platform. We will now examine how alternate configurations will affect how Token-Based Authentication interacts with those URLs.

**Note:** Each row in the following table represents a separate Token-Based Authentication configuration.

| Secured Location | Description |
|---|---|
| / | A valid token is required for all four URLs. |
| /Secure/Data | A valid token is only required for the second URL. |
| /MyServer | A valid token is required for the third and fourth URLs. |
| /MyServer/Secure | A valid token is required for the third and fourth URLs. |

## Flash Media Streaming Platform

The Flash Media Streaming platform provides two different mechanisms for delivering content, which are Live StreamCast and On-Demand. Although content generated for the Live StreamCast and On-Demand services are served from different servers, setting up a location that will require authentication can potentially protect streams requested from both of these services.

**Tip:** For detailed scenarios that describe how to secure a live event or on-demand content, please refer to **Appendix B: Flash Content Security Scenarios**.

**Note:** Wildcard characters (e.g., *) are not supported when setting up the folders or live ingestion points that will be secured.

### Live StreamCast

Keep the following items in mind when protecting your live streams:

- The publishing point URL is used to determine whether a live stream will be protected by Token-Based Authentication.
- Live streams cannot be protected on a per stream basis. Flash content can only be protected by securing a folder path.
- 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.

#### Securing Live Streams (Flash Media Streaming Platform)

In order to protect a live stream, you will need to set the publishing point location as a secure location. The publishing point location is the location referenced by the publishing point URL in the encoder. This relative path starts directly after the content access point for all CDN and edge CNAME URLs. This means that you have to specify the entire path after the content access point (e.g., /20*xxxx*). Additionally, the path to a protected folder always starts with a forward slash (/).

**Sample Configuration A**

If an encoder is set to publish to:

- rtmp://fso.lax.*xxxx*.edgecastcdn.net/20*xxxx*

Then the following location should be secured in the **Directories to Authenticate** option:

- /

**Reminder:** Securing the root folder (i.e., /) will require a valid token for all live streams and on-demand content.

**Sample Configuration B**

If an encoder is set to publish to:

- rtmp://fso.lax.*xxxx*.edgecastcdn.net/20*xxxx*/2012/Videos

Then one of the following locations should be secured in the **Directories to Authenticate** option:

- /
- /2012
- /2012/Videos

**Reminder:** All three of the above configurations can protect the specified live event due to the fact that security is applied recursively to all of the locations specified under the **Directories to Authentication** section.

## On-Demand Content

Keep the following items in mind when protecting your on-demand content:

- Unlike Live StreamCast, the storage location of your on-demand content is used to determine whether it will be protected by Token-Based Authentication.
- On-demand content cannot be protected on a per file basis. Flash content can only be protected by securing a folder path.
- Securing the root folder will require a valid token for all live and on-demand content.

### Securing On-Demand Content

In order to protect on-demand content, you will need to set the storage location of the desired on-demand content as a secure location. When specifying the location to be authenticated, you should only include the relative path that appears after the content access point. This configuration is required regardless of whether you plan on using a CDN or edge CNAME URL to access your content. Failure to include the entire path after the content access point will prevent that location from being protected by Token-Based Authentication.

**Note:** Although an edge CNAME URL does not include the name of a customer origin server and may not include the path to the requested content, it will be treated as if the corresponding CDN URL had been used. As a result, when securing such a location you must specify the name of the customer origin server followed by the relative path to the desired folder (e.g., /MyCustomerOrigin/Presentations/2012).

## Windows Media Streaming Platform

The Windows Media Streaming platform allows you to stream live events and on-demand content. Although these services behave differently, securing a location will protect streams requested from both of these services.

**Note:** Wildcard characters (e.g., *) are not supported when setting up the folders or live ingestion points that will be secured.

### Live Streams

Your live streams can be secured by either securing the root folder or by adding each live ingestion point that should be protected. If you secure the root folder (/), then a valid token will be required when connecting to any live or on-demand Windows Media stream. If you prefer to secure individual live ingestion points, then you should make sure to only specify the name of the desired live ingestion point when adding a new authentication location (e.g., MyLiveIngestionPoint).

**Important:** If you would like to secure live ingestion points individually, then you should not prepend a forward slash (/). Simply add the name as it appears under the **Name** column of the **Publishing Points** page, which can be found on the **Windows** tab in the MCC.

**Important:** If you are streaming content that has been protected by Token-Based Authentication, then you will need to verify that a token is appended to the live playback URL every time that it is referenced in your code.

### On-Demand Content

On-demand content can be protected by either securing the root folder or the path to the folder where the desired on-demand content is stored. If you choose to secure content in a particular location, then you will need to specify the relative path from the root folder of the CDN origin server. A forward slash (/) is used to represent the root folder. If you secure the root folder (/), then a valid token will be required when connecting to any live or on-demand Windows Media stream.

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

This section illustrates how different URLs interact with Token-Based Authentication. The URLs that will be studied are listed below.

1. mms://wms.0001.edgecastcdn.net/000001/ Secure/Presentation01.wmv
2. mms://wms.0001.edgecastcdn.net/000001/ Secure/2012/Presentation01.wmv?c1019f8a6942b46a1ce679e66cd579767

We will now examine the effect of securing a location called "/Secure" will have on the above URLs.

1. The first URL points to an asset stored in a folder called "Secure" on a CDN origin server. Since the asset is stored in a folder protected by Token-Based Authentication, it requires a token. Since a token was not specified for this request, the asset will not be served to the client.
2. The second URL points to an asset stored on a CDN origin server. Since this asset is located in a subfolder of a protected folder, it is also protected by Token-Based Authentication. The requested asset will be delivered to the client, as long as the token is valid and the user requesting it meets the requirements specified in the provided token.

We have just examined how different URLs would be affected when the "/Secure" location was secured for the Windows Media Streaming platform. We will now examine how alternate configurations will affect on-demand content streamed for those same URLs.

**Note:** Each row in the following table represents a separate Token-Based Authentication configuration.

| Name | Description |
|---|---|
| / | All requests will require a valid token. |
| /Secure/2012 | Only the second URL will require a valid token. |
| /Secure/2012/Presentation01 | None of the requested assets will be protected by Token-Based Authentication. |

## Authentication Folder Administration

The locations that will be secured through Token-Based Authentication can be administered on a per platform basis. You can choose to add, modify, or delete each location from the **Token Auth** page for the desired platform.

**Note:** It may take up to an hour for the creation, modification, or deletion of a location secured by Token-Based Authentication to take effect.

**To specify a new location that will be secured by Token-Based Authentication**

1. From the MCC, navigate to the **Token Auth** page for the desired platform.
2. From the **Directories to Authenticate** section, type the relative path to the folder whose contents you would like to protect in the **New** option.
3. Click **Add**.

**To modify a location that will be secured by Token-Based Authentication**

1. From the MCC, navigate to the **Token Auth** page for the desired platform.
2. From the **Directories to Authenticate** section, click ✏. The desired relative path will now be displayed in an edit box.
3. Modify the relative path to point to the folder whose contents you would like to protect.
4. Click 💾.

**To delete a location that will be secured by Token-Based Authentication**

1. From the MCC, navigate to the **Token Auth** page for the desired platform.
2. From the **Directories to Authenticate** section, click ❌ next to the relative path that you would like to delete.
3. When prompted, click **OK** to confirm that the relative path will be deleted.

# Protecting your Content by Request Type

Token-Based Authentication can be enabled or disabled based on the type of request that was received. HTTP Rules Engine, which must be purchased separately, provides this functionality. HTTP Rules Engine allows an administrator to set up rules that determine how requests that meet predefined criteria will be handled. For example, it can be configured to require a token value to be specified for all HTML assets that reside on a particular customer origin server. For detailed information on HTTP Rules Engine, please refer to the **HTTP Rules Engine Administration Guide**.

**Note:** The extent of HTTP Rules Engine functionality is not limited to determining whether a request will require Token-Based Authentication. There are a wide range of features that can be applied to a request that matches the criteria specified in a rule.

## Interaction with CDN Settings

HTTP Rules Engine both complements and overrides the default manner that our CDN handles requests for content. This means that a rule will only override your CDN configuration when it conflicts with the actions defined in that rule. This allows you to define a base configuration and then using HTTP Rules Engine to customize it to meet the specific needs of your organization.

For example, HTTP Rules Engine can be used to override the directories that have been secured by Token-Based Authentication for certain file types. You can create a rule that turns off Token-Based Authentication for all HTML, JavaScript, and CSS files. This type of rule ensures that those file types will not be protected by Token-Based Authentication regardless of where they are stored.

# Determining How to Protect Your Content

## Overview

There are many different requirements that you can set to control access to assets stored in protected folders. You can mix-and-match these requirements as required to secure your content. Adding, modifying, or removing requirements will simply create another unique token. Clients that use the older token will still be able to access your content as long as they still meet its requirements and the encryption key used to generate it is still active.

**Tip:** You can make tokens specific to a particular folder or asset. This will prevent that token from being reused to access other protected content. For more information, please refer to the **Preventing the Reuse of Token Values** section.

**Note:** If you would like to invalidate old tokens, then you will need to change your encryption key. For more information, please refer to the **Changing Your Encryption Key** section.

As previously noted, content in protected folders can only be accessed by appending a token value to the requested asset. Since each protected asset must be assigned a token, this allows you the flexibility to choose how requirements are applied to protected content. You can choose to apply the same set of requirements to all of your protected content or you can customize your requirements to fit the security needs of each asset.

### Setting Content Expiration Date

Time-sensitive content can be configured to only be available for a limited amount of time. Once the specified time frame for a token has expired, users will be denied access to that asset when requested using that token value.

The security parameter that controls time-based authentication is ec_expire. This parameter uses the number of seconds since Unix time (a.k.a. POSIX time or Unix epoch) to specify a date and time. Once the specified date and time has passed, requests that pass that token will be denied. Unix time starts on 1970-01-01 at 00:00:00 GMT.  For example, setting this parameter to "1356955200" would set the expiration date and time to 12/31/2012 12:00:00 GMT.

**Tip:** If you would like to set this parameter manually (rather than programmatically) or if you are trying to troubleshoot a particular token, then you can take advantage of one of the many websites that provide conversion to and from standard time conventions to Unix time.

## Allowing or Blocking Users by Country

You can choose to allow or block access to protected content based on the country from which the request originated. If you choose to allow customers by the country from which the request originated, then all countries that have not been specified will not have access to your content. On the other hand, if you choose to block by country, then all countries that have not been specifically blocked will continue to have access to your content, provided that they meet all other requirements specified by the token used to request the desired asset.

**Note:** If you would like to restrict access by country for the HTTP Large Object, HTTP Small Object, or the ADN platforms, then you have another option at your disposal. The option to filter by country is available from the **Country Filtering** page on the tab corresponding to the desired platform. Setting security from the **Country Filtering** page doesn't require the use of tokens, which speeds up your security implementation across your desired folders. For more information on country filtering options, please refer to the **HTTP Large Object**, **HTTP Small Object**, or the **ADN Administration Guide**.

The security parameter that allows access by country is called "ec_country_allow," while the one that denies access by country is called "ec_country_deny." The valid values for these parameters consist of any two-letter ISO 3166 country code. If you would like to specify more than one country per parameter, then simply separate each country code with a comma (e.g., US,GB,MX,FR). For a list of valid country codes, please refer to the **Appendix A: Country Codes**.

**Warning:** When specifying multiple country codes, make sure that you do not add a space along with the comma delimiter. Country codes that are preceded by a space will be excluded from a token's requirements.

**Note:** Although a typical configuration should not include both parameters, it is possible for a token to contain both of these requirements. In such a case, the Allow Country parameter (i.e., ec_country_allow) takes precedence over the Deny Country parameter (i.e., ec_country_deny).

**Note:** Country codes are case-insensitive.

## Preventing the Reuse of Token Values

Most security parameters generate tokens that are valid across all protected content. However, the Allow URL parameter (i.e., ec_url_allow) allows you to tailor your tokens to a particular asset or path. This parameter only validates requests that originate from certain URLs. The configuration for this parameter varies according to the platform through which your protected content will be accessed.

**Note:** The Allow URL parameter is not available for the Windows Media Streaming platform.

## Allow URL Parameter & HTTP-Based Platforms

This security parameter only verifies that the path to the requested object begins with the value assigned to it. This allows it the flexibility to validate the recursive contents of a folder or a specific asset. The starting point for this comparison occurs directly after the hostname specified in the URL. This occurs regardless of whether you are using a CDN or edge CNAME URL. For example, the Allow URL parameter would be compared against "/2012/06/Video.flv" for the following request URL: http://secure.mydomain.com/2012/06/Video.flv.

**Important:** A CDN or edge CNAME URL is case-sensitive. Please make sure to use the proper case when linking to CDN content or setting a value for the ec_url_allow parameter.

This parameter can also validate access to multiple folders or assets. This can be accomplished by separating each path with a comma (e.g., /000001/Folder1,/000001/Folder1/SubfolderA,/000001/Folder1/index.htm).

**Warning:** When specifying multiple assets or folders, make sure that you do not add a space along with the comma delimiter. Relative paths that are preceded by a space will be excluded from a token's requirements.

In order to demonstrate the proper syntax for this parameter, we have provided three sample URLs that point to a folder called "Secure." This folder has been secured with Token-Based Authentication.

1. http://wpc.0001.edgecastcdn.net/000001/Secure/index.html
2. http://wpc.0001.edgecastcdn.net/800001/MyServer/Secure/index.html
3. http://secure.server.com/index.html

**Note:** Although the above sample URLs are specific to the HTTP Large Object platform, the analysis of these URLs also applies to the HTTP Small Object and the ADN platforms.

We will now examine the base value that must be assigned to the ec_url_allow parameter to grant access to the above URLs. The first URL points to a CDN origin server. As such, the base value that you should assign to the ec_url_allow parameter is "/000001." If you would like to secure each asset individually, then you would set the ec_url_allow parameter to the desired asset, which in this case would be "/000001/Secure/index.html." If this same folder contained another asset called "Confidential.doc," then you could grant access to both assets by either generating a token for the parent folder, for each individual asset, or for both assets. The last scenario can be achieved by setting the ec_url_allow parameter to "/000001/Secure/index.html,/000001/Secure/Confidential.doc."

The second URL points to a customer origin server. The "MyServer" folder is the name assigned to the customer origin configuration for the server hosting your content. In this example, you would use "/800001/MyServer" as the base value for the ec_url_allow parameter.

The third example uses an edge CNAME in the URL. In this particular case, "secure.server.com" points to the same "Secure" folder used in the second example. The base value for the ec_url_allow parameter would be "/," since this example uses an edge CNAME (i.e., MyServer).

We will now use the base edge CNAME URL from the third example to demonstrate how access will be granted or denied based on tokens that take advantage of the ec_url_allow parameter. The following scenario assumes that the token used to request access has the following requirement:

- ec_url_allow=/Folder1/movie1,/Folder2

In this scenario, the following requests would be allowed:

- http:// secure.server.com/Folder1/movie1.flv
- http:// secure.server.com/Folder1/movie1.mpg
- http:// secure.server.com/Folder1/movie1/index.htm
- http:// secure.server.com/Folder2/film.mpg

The following requests would be denied:

- http:// secure.server.com/Folder1/movie2.flv
- http:// secure.server.com/Folder3

## Allow URL Parameter & Flash Media Streaming Platform

For each request, a comparison is performed between the value assigned to this parameter and the CDN URL path. This occurs regardless of whether a CDN or edge CNAME URL was used to request the secured content. The CDN URL path is the portion of the URL that appears directly after the CDN domain. If the CDN URL path begins with the specified value, then this token requirement will be satisfied. This allows it the flexibility to validate the recursive contents of a folder or a specific asset.

**Important:** A CDN URL is case-sensitive. In addition to making sure that your CDN URL uses the proper case, you should also ensure that the case matches when specifying a value for the ec_url_allow parameter.

**Important:** This parameter always compares URLs against the CDN URL path. This occurs regardless of whether you are using a CDN or edge CNAME URL. For example, if you are securing an asset stored on a CDN origin server and your account number is 0001, then the starting value for this parameter would be: /000001. The fact that the content access point (e.g., /000001) is not displayed for an edge CNAME URL does not affect this behavior.

This parameter can also validate access to multiple folders or assets. This can be accomplished by separating each path with a comma (e.g., /000001/Folder1,/000001/Folder1/SubfolderA,/000001/Folder1/movie.flv).

**Warning:** When specifying multiple assets or folders, make sure that you do not add a space along with the comma delimiter. Relative paths that are preceded by a space will be excluded from a token's requirements.

In order to demonstrate the proper syntax for this parameter, we have provided three sample URLs that point to a folder called "Secure." This folder has been secured with Token-Based Authentication.

- rtmp://fms.0001.edgecastcdn.net/000001/Secure/movie.flv
- rtmp://fms.0001.edgecastcdn.net/800001/MyServer/Secure/movie.flv
- rtmp://secure.server.com/movie.flv

We will now examine the starting value that must be assigned to ec_url_allow for the above URLs. The first URL points to a CDN origin server. As such, the base value that you should assign to the ec_url_allow parameter is "/000001." If you would like to secure each asset individually, then you would set the ec_url_allow parameter to the desired asset, which in this case would be "/000001/Secure/movie.flv." If this same folder contained another asset called "Confidential.mp4," then you could grant access to both assets by either generating a token for the parent folder (Secure), for each individual asset, or for both assets. The last scenario can be achieved by setting the ec_url_allow parameter to "/000001/Secure/movie.flv,/000001/Secure/Confidential.mp4."

The second URL points to a customer origin server. The "MyServer" folder is the name assigned to the customer origin configuration for the server hosting your content. In this example, you would use "/800001/MyServer" as the base value for the ec_url_allow parameter.

The third example uses an edge CNAME in the URL. In this particular case, "secure.server.com" points to the same "Secure" folder used in the second example. Since this example uses an edge CNAME (i.e., MyServer) that points to a subfolder of that server (i.e., Secure), you would use "/800001/MyServer/Secure" as the base value for the ec_url_allow parameter.

We will now use the base edge CNAME URL from the third example to demonstrate how access will be granted or denied based on tokens that take advantage of the ec_url_allow parameter. The following scenario assumes that the token used to request access has the following requirement:

ec_url_allow=/800001/MyServer/Secure/Folder1/movie1,/800001/MyServer/Secure/Folder2

In this scenario, the following requests would be allowed:

- rtmp://fms.0001.edgecastcdn.net/800001/MyServer/Secure/Folder1/movie1.flv
- rtmp://fms.0001.edgecastcdn.net/800001/MyServer/Secure/Folder2/movie1.mp4
- rtmp://secure.server.com/Folder1/movie1/video.mp4
- rtmp://secure.server.com/Folder2/film.f4v

The following requests would be denied:

- rtmp://secure.server.com/Folder1/movie2.flv
- rtmp://secure.server.com/Folder3/movie1.flv

## Allowing or Blocking Users by Host

You can choose to allow or block users based on the host requesting protected content. A host, which is reported by the Host request header field, identifies the hostname of the server from which the content was requested.

**Reminder:** Although these parameters can be used with the HTTP Large Object, HTTP Small Object, and the ADN platforms, they are not available from the **Encrypt Tool** section of the **Token Auth** page. However, you can still generate an encrypted token value by using the Token Generation application or by creating your own token generator.

The security parameter that allows access by host is called "ec_host_allow,**"** while the one that denies access by host is called "ec_host_deny." When specifying a hostname, you should not include the protocol (e.g., http://) or the port number (e.g., :100) associated with the host. If you would like to validate more than one host within a single parameter, you may do so by separating each one with a comma.

**Warning:** When specifying multiple hosts, make sure that you do not add a space along with the comma delimiter. Hostnames that are preceded by a space will be ignored.

**Note:** Although the Host request header field will include port information when a non-default port (e.g., www.domain.com:100) is used, it is ignored by this security parameter. This security parameter performs comparisons on the hostname without port information.

**Note:** Although a typical configuration should not include both parameters, it is possible for a token to contain both of these security parameters. In such a case, the ec_host_allow parameter takes precedence over the ec_host_deny parameter.

### Wildcard Matching for Subdomains (HTTP-Based Platforms)

If you would like to specify a wildcard hostname, then you may do so by specifying a host using this format: *.*Domain*. This type of configuration will match any host that contains the specified domain (e.g., www.domain.com, secure.domain.com, and videos.domain.com).

**Note:** The asterisk (*) character only acts as a wildcard character when it occurs as the first character in the specified hostname.

**Allow/Deny Host Examples**

We will now use a sample URL to demonstrate how access will be granted or denied based on tokens that take advantage of the ec_host_allow parameter. The following scenario assumes that the token used to request access has the following requirement:

- ec_host_allow=www.server1.com,data.server1.com,*.server2.com

In this scenario, the following hosts would be allowed:

- www.server1.com
- data.server1.com
- secure.server2.com
- en.secure.server2.com

The following requests would be denied:

- secure.server1.com
- server2.com

The ec_host_deny parameter works in the same way. The following scenario assumes that the token used to request access has the following requirement:

- ec_host_deny=www.server1.com,data.server1.com,*.server2.com

In this scenario, the following hosts would be allowed:

- secure.server1.com
- server2.com

Requests with the following hosts would be denied:

- www.server1.com
- data.server1.com
- secure.server2.com
- en.secure.server2.com

## Allowing or Blocking Users by Referrer

You can choose to allow or block users based on the referrer used to access protected content. A referrer is the URL for the web page from which the link was followed. The security parameter that allows access by referrer is called "ec_ref_allow," while the one that denies access by referrer is called "ec_ref_deny." When specifying a referrer, you should not include the protocol portion of the URL (e.g., http://). If a referrer's URL path begins with the specified value, then this token requirement will be satisfied. This allows the flexibility to validate a domain and/or a particular path on that domain. Additionally, if you would like to validate more than one referrer within a single parameter, you may do so by separating each one with a comma.

**Warning:** When specifying multiple referrers, make sure that you do not add a space along with the comma delimiter. Referrers that are preceded by a space will be ignored.

**Important:** The referrer that is passed for the Windows Media Streaming platform is different from the one used by the other platforms. It is only passed when the media player is embedded on a web page. In such a case, the URL for that web page is the referrer that will be reported. All other configurations will result in a blank referrer.

**Note:** Although a typical configuration should not include both parameters, it is possible for a token to contain both of these security parameters. In such a case, the ec_ref_allow parameter takes precedence over the ec_ref_deny parameter.

**Reminder:** Although these parameters can be used with the Windows Media Streaming platform, they are not available from the **Encrypt Tool** section of the **Token Auth** page. However, you can still generate an encrypted token value by using the Token Generation application or by creating your own token generator.

### Wildcard Matching for Subdomains (HTTP-Based Platforms)

If you are protecting content for either the HTTP Large Object, HTTP Small Object, or the ADN platform, then you may use a single asterisk (*) as a wildcard at the beginning of the assigned parameter value. Parameters configured in this way will match zero or more characters for the subdomain portion of the URL (e.g., secure in secure.domain.com). A wildcard will not match forward slashes (/), nor can it be used to match characters in other portions of the URL.

### Handling Missing or Blank Referrers

Some browsers can be configured to not send referrer information. By default, the ec_ref_allow parameter will block these requests, since they do not match the specified criteria. Likewise, the default behavior of ec_ref_deny is to allow these requests. If you would like to change the default way in which blank or missing referrers are handled, then you should assign either a "Missing" or a blank value to the desired parameter.

All of the following sample values will grant access for requests with blank or missing referrers.

- ec_ref_allow=www.server1.com/Folder1/movie1,data.server1.com,MISSING
- ec_ref_allow=www.server1.com/Folder1/movie1,data.server1.com,
- ec_ref_deny= www.server1.com/Folder1/movie1,data.server1.com

**Note:** The trailing comma in the second example allows blank or missing referrers.

All of the following sample values will deny access for requests with blank or missing referrers.

- ec_ref_allow=www.server1.com/Folder1/movie1,data.server1.com
- ec_ref_deny= www.server1.com/Folder1/movie1,data.server1.com,MISSING

- ec_ref_deny= www.server1.com/Folder1/movie1,data.server1.com,

**Referrer Examples**

We will now use a sample URL to demonstrate how access will be granted or denied based on tokens that take advantage of the ec_ref_allow parameter. The following scenario assumes that the token used to request access has the following requirement:

- ec_ref_allow=www.server1.com/Folder1/movie1,data.server1.com,*.server2.com

**Reminder:** Wildcards are only supported for this security parameter on the HTTP Large Object, HTTP Small Object, or the ADN platforms.

In this scenario, requests with the following referrers would be allowed:

- http:// www.server1.com/Folder1/movie1.flv
- http:// www.server1.com/Folder1/movie1.mpg
- http:// www.server1.com/Folder1/movie1/index.htm
- https:// data.server1.com/Folder2/movie123.mpg
- https://secure.server2.com/index.html
- https://en.secure.server2.com/index.html

Requests with the following referrers would be denied:

- [Blank or not provided]
- http://www.server1.com/
- http:// secure.server1.com/Folder1/movie1.flv
- http://server2.com/index.html
- http://domain.com/secure.server2.com/index.html

The ec_ref_deny parameter works in the same way. The following scenario assumes that the token used to request access has the following requirement:

- ec_ref_deny=www.server1.com/Folder1/movie1,data.server1.com,*.server2.com

**Reminder:** Wildcards are only supported for this security parameter on the HTTP Large Object, HTTP Small Object, or the ADN platforms.

In this scenario, requests with the following referrers would be allowed:

- [Blank or not provided]
- http://www.server1.com/
- http:// secure.server1.com/Folder1/movie1.flv

- http://server2.com/index.html
- http://domain.com/secure.server2.com/index.html

Requests with the following referrers would be denied:

- http:// www.server1.com/Folder1/movie1.flv
- http:// www.server1.com/Folder1/movie1/index.htm
- https:// data.server1.com/Folder2/movie123.mpg
- https://secure.server2.com/index.html
- https://en.secure.server2.com/index.html

## Allowing Users by IP address

You can choose to only allow requests that originate from a specific IP address access to content stored in a protected folder. All other IP addresses will be denied access. This can be accomplished through the **ec_clientip** parameter. This parameter uses standard IPv4 notation (e.g., 100.10.123.45).

## Allowing or Blocking Users by Protocol

You can choose to allow or block users depending on the protocol used to request the desired content. The security parameter used to allow access by protocol is called "ec_proto_allow," while the one that is used to deny access by protocol is called "ec_proto_deny." The only valid values for these parameters are "http" and "https." You can choose to allow or deny both parameters by setting the desired parameter to "http,https."

**Important:** Keep in mind that the values specified for this parameter are case-sensitive. Make sure to specify the protocol in lower-case letters (e.g., http or https).

**Note:** The Allow Protocol and the Deny Protocol security parameters are not available for Flash Media Streaming or the Windows Media Streaming platforms. Additionally, parameter values for the protocols associated with these platforms are not available. This means that you cannot use these parameters to prevent access to assets from the Windows Media Streaming or the Flash Media Streaming platform.

**Note:** Although a typical configuration should not include both parameters, it is possible for a token to contain both of these requirements. In such a case, the ec_proto_allow parameter takes precedence over the ec_proto_deny parameter.

## Preventing Changes to Bandwidth Throttling Settings

Bandwidth throttling provides the ability to limit the rate at which a user can download an asset. This capability is controlled by the ec_rate and ec_prebuf parameters. Typically, these parameters are specified as query string parameters. However, if you would like to encrypt these parameters, then you will need to generate a token value that includes the desired values for these parameters. By preventing a user from altering the values assigned to these parameters, you can ensure that data downloads are throttled to the desired level.

**Note:** Both of these parameters are not available from the **Encrypt Tool** section of the **Token Auth** page. However, you can still generate an encrypted token value by using the Token Generation application or by creating your own token generator.

**Note:** Bandwidth throttling is only available for the HTTP Large Object platform. For more information, please refer to the **HTTP Large Object Administration Guide**.

# Generating Tokens

## Overview

A token value is required to access all content protected by Token-Based Authentication. Before you can assign a token value to a link, you will need to generate it with the desired security requirements for the asset in question. When generating a token, keep in mind that there is no limit to the number of security parameters that can be combined. In other words, a token value can consist of a single or multiple parameters. Additionally, you should also keep in mind that certain parameters support multiple values. This permits a lot of flexibility when determining the security requirements for your protected content.

**Important:** Although there is no limit to the number of security parameters that can be combined to form a token, there is a limit of 512 characters for the total length of a token. In order to prevent your clients from being inadvertently denied access to your content, please ensure that your token values never exceed 512 characters.

**Note:** Regardless of the manner in which you generate tokens, keep in mind that this process will not affect your Token-Based Authentication configuration in any way. Additionally, there is no limit to the number of token values that you can generate for a particular encryption key.

We offer two direct ways to generate a token, which are through the MCC and using the Token Generation application (ec_encrypt.exe). Additionally, we also provide source code for a token generator that allows you to incorporate token generation capabilities into your code. All three methods for generating tokens are explained below.

**Reminder:** Token values are not inherently folder or platform-specific. This means that a user that satisfies a token’s requirements can use that token to retrieve content from any protected folder that has been associated with the encryption key used to generate it, as long as the token’s requirements are not specific to that path or asset. This type of configuration makes it possible to gain access to protected content from various folders across different platforms.

### Manually Generating a Token

An individual token value can be generated through the MCC. This can be accomplished through the **Encrypt Tool** section of the **Token Auth** page. The sole purpose of this section is to generate a token value based on either the primary or backup key. Once you have specified values for each desired parameter, simply select the desired key, and then click **Encrypt**. A token specific to the selected key will then be generated. This token value will appear next to the **Generated Token** label. You may then append a question mark and this token value to the desired request.

## Using Our Token Generation Application

Another way to generate token values is by downloading our Token Generation application (ec_encrypt). By offering an executable through which you can generate tokens, you can use a script to generate a token value and then use that value when creating links to protected content. We provide both a Windows (ec_encrypt.exe) and a Linux (ec_encrypt) version of this executable.

**Note:** The Windows version of our executable requires Blowfish.dll and Blowfish.xml. Please make sure that these assets are stored in the same folder as the encryption executable.

The proper syntax for specifying a single parameter with ec_encrypt is the following:

ec_encrypt.exe *KeyName* "*parameter=value*"

The proper syntax for specifying multiple parameters is to use an ampersand (&) between parameters. This can be seen in the following syntax example:

ec_encrypt.exe *KeyName* "*parameter1=value*&*parameter2=value1*,*value2*"

For example, if you wanted to generate a token that meets the following requirements:

- Uses an encryption key called "MyKey."
- Expires on 12/31/2012 12:00:00.
- Only allows access to North American countries.
- Only allows referrers from "TrustedDomain.com."

Then you would use the following syntax:

ec_encrypt.exe MyKey "ec_expire=1356955200&ec_country_allow=US,CA,MX&ec_ref_allow=*.TrustedDomain.com"

The token value associated with this security configuration would be:

1ea46ba396e88f03a9f6b7b06ab32d2f6acf8cd3f674597be08059b7655bd77e35cca67cabfd149b46af1c37a6c8c790f2ecbac7d84ee6a8cfc88409f12ba7635f1123a5e9e71ee92bae503dbd7c7131429b79f9809dbc3df2d5e46328

You would then append this token value to your protected content as can be seen below:

http://secure.server.com/MyProtectedAsset.html?1ea46ba396e88f03a9f6b7b06ab32d2f6acf8cd3f674597be08059b7655bd77e35cca67cabfd149b46af1c37a6c8c790f2ecbac7d84ee6a8cfc88409f12ba7635f1123a5e9e71ee92bae503dbd7c7131429b79f9809dbc3df2d5e46328

**Tip:** The proper syntax for your desired security configuration can be viewed after manually generating a token through the **Encrypt Tool** section of the **Token Auth** page, which can be found in the MCC. Once you have encrypted a token value with the desired requirements, the exact syntax that should be used to generate that token will appear next to the **Token Generator Call** label.

### Building a Token Generator

We provide C, C#, and PHP source code that can be used to incorporate our token generating capabilities with your code. This code, as well as information on implementation, is available upon request. Please contact your CDN account manager for more information.

## Decrypting an Existing Token

If you know the exact encryption key that was used for a particular token, then you can decrypt it. Decrypting an existing token allows you to view its security requirements. If you suspect that a particular client is having trouble viewing your protected content, you can decrypt his/her token to discover which security requirement is not being met.

A token can be decrypted from the **Decrypt Tool** section of the **Token Auth** page. Once you have copied the token value to the **Token To Decrypt** option, simply select the appropriate encryption key, and then click **Decrypt**. The security requirements for that token will appear next to the **Original Parameters** label.

**Note:** If the decryption tool indicates that it was unable to decrypt the specified token value, then you have selected an invalid encryption key. Keep in mind that the token value may have been generated with an outdated encryption key. If you suspect that you know the encryption key that was used for that token value, then you can temporarily set the backup encryption key to that value, decrypt the token value, and then clear the backup key.

# Providing Access to Protected Content

## Overview

Providing access to protected content is quite simple. It is just a matter of appending a question mark and an appropriate token value to the name of the asset being requested. Below you will find examples for each platform.

**Note:** Please refer to the **Generating Tokens** section for information on how to generate a token.

### HTTP Large Object Example

The sample code excerpt provided below demonstrates how to provide access for content requested through a CDN and an edge CNAME URL on the HTTP Large Object platform.

```
<a href="http://wpc.0001.edgecastcdn.net/000001/secure/index.html?
c1019f8a6942b46a1ce679e168d5797670f3ee7e39068054ee4534d8a5a859dc06&user=Joe">

<img src="http://images.mydomain.com/images/myimage.jpg?
c1019f8a6942b46a1ce679e168d5797670f3ee7e39068054ee4534d8a5a859dc06">
```

The first sample CDN URL contains a custom query string parameter called "user." As you can tell, custom query string parameters can be appended to the URL through the use of an ampersand. The second sample URL demonstrates how a token value can be specified with an edge CNAME URL.

### HTTP Small Object Example

The sample code excerpt provided below demonstrates how to provide access for content requested through a CDN and an edge CNAME URL on the HTTP Large Object platform.

```
<a href="http://wac.0001.edgecastcdn.net/000001/secure/index.html?
c1019f8a6942b46a1ce679e168d5797670f3ee7e39068054ee4534d8a5a859dc06&user=Joe">

<img src="images.mydomain.com/images/myimage.jpg?
c1019f8a6942b46a1ce679e168d5797670f3ee7e39068054ee4534d8a5a859dc06">
```

The first sample CDN URL contains a custom query string parameter called "user." As you can tell, custom query string parameters can be appended to the URL through the use of an ampersand. The second sample URL demonstrates how a token value can be specified with an edge CNAME URL.

## ADN Example

The sample code excerpt provided below demonstrates how to provide access for content requested through a CDN and an edge CNAME URL on the ADN platform. Notice how a query string parameter called "user" is appended to the URL through the use of an ampersand.

```
<a href="http://adn.0001.edgecastcdn.net/000001/secure/default.php?
c1019f8a6942b46a1ce679e168d5797670f3ee7e39068054ee4534d8a5a859dc06&user=Joe">

<a href="http://dynamic.mydomain.com/secure/default.php?
c1019f8a6942b46a1ce679e168d5797670f3ee7e39068054ee4534d8a5a859dc06&user=Joe">
```

## Flash Media Streaming Example

Although the URLs used for the Live StreamCast and On-Demand services are different, the syntax for specifying a token value is the same. Sample code excerpts are provided for each service below.

**Live StreamCast:**

```
<script type="text/javascript">
        jwplayer("container").setup({
        flashplayer: "player.swf",
        height: 270,
        width: 480,
        file: "StreamName?c1019f8a6942b46a1ce679e168d5797670f3ee7e39068054ee4534d8a5",
        provider: "rtmp",
        streamer: "rtmp://fml.0001.edgecastcdn.net/200001/My_Folder",
        'rtmp.subscribe':'true'
        });
</script>
```

**On-Demand:**

```
<script type="text/javascript">
        jwplayer("container").setup({
        flashplayer: "player.swf",
        height: 270,
        width: 480,
        file: "Presentation01.flv?c1019f8a6942b46a1ce679e168d5797670f3ee7e39068054e4",
        provider: "rtmp",
        streamer: "rtmp://fms.0001.edgecastcdn.net/000001/My_Folder"
        });
</script>
```

### Windows Media Streaming Example

Although the URLs used for a live event or on-demand content are different, the syntax for specifying a token value is the same. Sample code excerpts are provided for each service below.

**Live Streaming:**

```
src="mms://wms.0001.edgecastcdn.net/200001/MyStream?c1019f8a6942b46a1ce679e168d5797670f
3ee7e39068054ee4534d8a5a859dc06"
```

**On-Demand:**

```
src="mms://wms.0001.edgecastcdn.net/000001/My_Folder/Presentation01.wmv?c1019f8a6942b46
a1ce679e168d5797670f3ee7e39068054ee4534d8a5a859dc06"
```

## Redirecting Unauthorized Users

By default, a user that does not meet the minimum security requirements for protected content will view a web page with a 403 Forbidden status code. If the protected content is accessed through either the HTTP Large Object, HTTP Small Object, or the ADN platform, then you can customize the status code that is returned or even redirect users to another web page.

The available alternative response codes are listed below.

| Response Code | Response Name | Description |
|---|---|---|
| 301 | Moved Permanently | This status code redirects unauthorized users to the URL specified in the Location header. |
| 302 | Found | This status code redirects unauthorized users to the URL specified in the Location header. This status code is the industry standard method of performing a redirect. |
| 307 | Temporary Redirect | This status code redirects unauthorized users to the URL specified in the Location header. |
| 403 | Forbidden | This is the standard 403 Forbidden status message that an unauthorized user will see when trying to access protected content. |
| 404 | File Not Found | This status code indicates that the HTTP client was able to communicate with the server, but the specified asset was not found. |

**To redirect unauthorized users to a user-friendly error page (recommended configuration)**

1. From the MCC, navigate to the **Token Auth** page for the desired platform.
2. From the **Custom Denial Handling** section, select "302" from the **Response Code** option. The **Header Name** option should automatically be set to "Location."
3. Make sure that the **Enabled** option is marked.
4. In the **Header Value** option, type the full URL to the user-friendly error page (e.g., http://www.server.com/PurchaseContent.aspx).
5. Click **Save**.

**Note:** The Location header URL can reside on any domain. It does not have to be hosted by our CDN.

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

# Quick Reference

## Security Parameters

This section provides a brief description for each available security parameter. For a detailed explanation of a particular parameter, please refer to the **Determining How to Protect your Content** section in the previous chapter.

| Parameter | Description |
|---|---|
| ec_clientip | Limits connections to requests originating from a specific IP address. This parameter uses standard IPv4 notation.<br>**Sample value:** 111.11.111.11<br>This example will only serve an asset to a client with an IP address of 111.11.111.11. |
| ec_country_allow | Defines the set of countries that will be allowed access to an asset through this token value. Acceptable values for this parameter consist of ISO 3166 country codes. Multiple country codes may be specified by separating them with a comma. Please refer to the **Appendix A: Country Codes** for a country code listing.<br>**Sample value:** US<br>This example will deny all requests that do not originate from the United States. |
| ec_country_deny | Denies requests from one or more countries. Acceptable values for this parameter consist of ISO 3166 country codes. Specify multiple country codes by separating each code with a comma. Please refer to the **Appendix A: Country Codes** for a country code listing.<br>**Sample value:** US,CA<br>This example will deny all requests that originate from the United States and Canada. |
| ec_expire | Sets an expiration date and time (GMT) for the token value. Set this parameter to the number of seconds that will pass from Unix time to the expiration date.<br>**Sample value:** 1356955200<br>This example will set the expiration date and time to 12/31/2012 12:00:00 GMT. |

| Parameter | Description |
|---|---|
| ec_host_allow | Defines the set of hosts through which the requesting client can gain access to an asset. This parameter should not include the protocol portion of the desired URL (e.g., http://). A comparison will be made against the value specified in the Host request header. If the hostname matches a specified value, then the requester will be allowed access. Specify multiple hosts by separating each one with a comma.<br>**Sample value:** server1.com,*.server2.com<br>This example will allow access to the following hosts:<br>• server1.com<br>• secure.server2.com |
| ec_host_deny | Defines the set of hosts for which the requesting client will be denied access to an asset. This parameter should not include the protocol portion of the desired URL (e.g., http://). A comparison will be made against the value specified in the Host request header. If the hostname matches a specified value, then the requester will be denied access. Specify multiple hosts by separating each one with a comma.<br>**Sample value:** server1.com,*.server2.com<br>This example will deny access to the following hosts:<br>• server1.com<br>• secure.server2.com |
| ec_proto_allow | Defines the protocol that can be used to retrieve an asset. Acceptable values for this parameter are "http" and "https."<br>**Sample value:** https<br>This example will only allow access to URLs that use the https protocol. |
| ec_proto_deny | Defines the protocol that cannot be used to retrieve an asset. Acceptable values for this parameter are "http" and "https."<br>**Sample value:** http<br>This example will deny access to URLs that use the http protocol. |

| Parameter | Description |
|---|---|
| ec_ref_allow | Defines the set of referrers through which the requesting client can gain access to an asset. This parameter should not include the protocol portion of the desired URL (e.g., http://). A comparison will be made against the value specified in the Referer header. If the starting characters in the referrer match a specified value, then the requester will be allowed access. Specify multiple referrers by separating each one with a comma.<br>**Sample value:** server1.com/obj1,*.server2.com<br>This example will allow access to the following referrers:<br>• server1.com/obj1/index.htm<br>• server1.com/obj1.html<br>• secure.server2.com/2012/Graph.xml |
| ec_ref_deny | Defines the set of referrers for which the requesting client will be denied access to an asset. This parameter should not include the protocol portion of the desired URL (e.g., http://). A comparison will be made against the value specified in the Referer header. If the starting characters in the referrer match a specified value, then the requester will be denied access. Specify multiple referrers by separating each one with a comma.<br>**Sample value:** server1.com/obj1,*.server2.com<br>This example will deny access to the following referrers:<br>• server1.com/obj1/index.htm<br>• server1.com/obj1.html<br>• secure.server2.com/2012/Graph.xml |
| ec_url_allow | Links a URL path to a token. Only requests that start with the specified URL path will be allowed access. This parameter should not include the protocol and domain portions of the desired URL (e.g., http://www.domain.com).<br>**Sample value:** /000001/dir1/movie1,/000001/dir2<br>Assuming that the above value was configured for the HTTP Large Object platform, this example will allow access for the following requests:<br>• http:// wpc.*xxxx*.edgecastcdn.net/00*xxxx*/dir1/movie1.flv<br>• http:// wpc.*xxxx*.edgecastcdn.net/00*xxxx*/dir1/movie1.mpg<br>• http:// wpc.*xxxx*.edgecastcdn.net/00*xxxx*/dir1/movie1/index.htm<br>• http:// wpc.*xxxx*.edgecastcdn.net/00*xxxx*/dir2/movie123.mpg |

# Bandwidth Throttling Reference

This section provides a brief description for each bandwidth throttling parameter that can be encrypted using Token-Based Authentication. For a detailed explanation of a particular parameter, please refer to the **Bandwidth Throttling** chapter in the **HTTP Large Object Administration Guide**.

**Note:** Bandwidth throttling parameters are only supported on the HTTP Large Object platform.

| Name | Description |
|---|---|
| ec_prebuf | This parameter determines how much data can be downloaded before the ec_rate parameter takes effect. Although this parameter is specified in seconds, the actual amount of data that can be buffered is calculated by multiplying the specified value by ec_rate.<br>**Sample value:** 10<br>Assuming that ec_rate has been set to 64, this example will allow a user to download 640 KB before bandwidth throttling will take effect. |
| ec_rate | This parameter limits the rate (KBps) at which clients will be able to download the specified asset.<br>**Sample value:** 64<br>This example will limit a user's download speed to 64 KBps. |

# Appendix A

## Country Codes (ISO 3166)

This section provides a list of country codes that are supported by the **ec_country_allow** and **ec_country_deny** security parameters. These country codes follow the ISO 3166 country code specification.

**Reminder:** Country codes are case-insensitive.

| Code | Country |
|---|---|
| AF | Afghanistan |
| AL | Albania |
| DZ | Algeria |
| AS | American Samoa |
| AD | Andorra |
| AO | Angola |
| AI | Anguilla |
| AQ | Antarctica |
| AG | Antigua and Barbuda |
| AR | Argentina |
| AM | Armenia |
| AW | Aruba |
| AP | Asia/Pacific Region |
| AU | Australia |
| AT | Austria |
| AZ | Azerbaijan |
| BS | Bahamas |
| BH | Bahrain |
| BD | Bangladesh |
| BB | Barbados |
| BY | Belarus |

| Code | Country |
|---|---|
| BE | Belgium |
| BZ | Belize |
| BJ | Benin |
| BM | Bermuda |
| BT | Bhutan |
| BO | Bolivia |
| BA | Bosnia and Herzegovina |
| BW | Botswana |
| BV | Bouvet Island |
| BR | Brazil |
| IO | British Indian Ocean Territory |
| BN | Brunei Darussalam |
| BG | Bulgaria |
| BF | Burkina Faso |
| BI | Burundi |
| KH | Cambodia |
| CM | Cameroon |
| CA | Canada |
| CV | Cape Verde |
| KY | Cayman Islands |
| CF | Central African Republic |
| TD | Chad |
| CL | Chile |
| CN | China |
| CX | Christmas Island |
| CC | Cocos (Keeling) Islands |
| CO | Colombia |
| KM | Comoros |
| CG | Congo |
| CD | Congo, The Democratic Republic of the |
| CK | Cook Islands |

| Code | Country |
|---|---|
| CR | Costa Rica |
| CI | Cote d'Ivoire |
| HR | Croatia |
| CU | Cuba |
| CY | Cyprus |
| CZ | Czech Republic |
| DK | Denmark |
| DJ | Djibouti |
| DM | Dominica |
| DO | Dominican Republic |
| EC | Ecuador |
| EG | Egypt |
| SV | El Salvador |
| GQ | Equatorial Guinea |
| ER | Eritrea |
| EE | Estonia |
| ET | Ethiopia |
| EU | Europe |
| FK | Falkland Islands (Malvinas) |
| FO | Faroe Islands |
| FJ | Fiji |
| FI | Finland |
| FR | France |
| GF | French Guiana |
| PF | French Polynesia |
| TF | French Southern Territories |
| GA | Gabon |
| GM | Gambia |
| GE | Georgia |
| DE | Germany |
| GG | Guernsey |

| Code | Country |
|---|---|
| GH | Ghana |
| GI | Gibraltar |
| GR | Greece |
| GL | Greenland |
| GD | Grenada |
| GP | Guadeloupe |
| GU | Guam |
| GT | Guatemala |
| GN | Guinea |
| GW | Guinea-Bissau |
| GY | Guyana |
| HT | Haiti |
| HM | Heard Island and McDonald Islands |
| VA | Holy See (Vatican City State) |
| HN | Honduras |
| HK | Hong Kong |
| HU | Hungary |
| IS | Iceland |
| IM | Isle of Man |
| IN | India |
| ID | Indonesia |
| IR | Iran, Islamic Republic of |
| IQ | Iraq |
| IE | Ireland |
| IL | Israel |
| IT | Italy |
| JE | Jersey |
| JM | Jamaica |
| JP | Japan |
| JO | Jordan |
| KZ | Kazakhstan |

| Code | Country |
|---|---|
| KE | Kenya |
| KI | Kiribati |
| KP | Korea, Democratic People's Republic of |
| KR | Korea, Republic of |
| KW | Kuwait |
| KG | Kyrgyzstan |
| LA | Lao People's Democratic Republic |
| LV | Latvia |
| LB | Lebanon |
| LS | Lesotho |
| LR | Liberia |
| LY | Libyan Arab Jamahiriya |
| LI | Liechtenstein |
| LT | Lithuania |
| LU | Luxembourg |
| MO | Macao |
| MK | Macedonia |
| MG | Madagascar |
| MW | Malawi |
| MY | Malaysia |
| MV | Maldives |
| ML | Mali |
| MT | Malta |
| MH | Marshall Islands |
| MQ | Martinique |
| MR | Mauritania |
| MU | Mauritius |
| YT | Mayotte |
| MX | Mexico |
| FM | Micronesia, Federated States of |
| MD | Moldova, Republic of |

| Code | Country |
|---|---|
| MC | Monaco |
| MN | Mongolia |
| ME | Montenegro |
| MS | Montserrat |
| MA | Morocco |
| MZ | Mozambique |
| MM | Myanmar |
| NA | Namibia |
| NR | Nauru |
| NP | Nepal |
| NL | Netherlands |
| AN | Netherlands Antilles |
| NC | New Caledonia |
| NZ | New Zealand |
| NI | Nicaragua |
| NE | Niger |
| NG | Nigeria |
| NU | Niue |
| NF | Norfolk Island |
| MP | Northern Mariana Islands |
| NO | Norway |
| OM | Oman |
| PK | Pakistan |
| PW | Palau |
| PS | Palestinian Territory |
| PA | Panama |
| PG | Papua New Guinea |
| PY | Paraguay |
| PE | Peru |
| PH | Philippines |
| PL | Poland |

| Code | Country |
|---|---|
| PT | Portugal |
| PR | Puerto Rico |
| QA | Qatar |
| RE | Reunion |
| RO | Romania |
| RU | Russian Federation |
| RW | Rwanda |
| SH | Saint Helena |
| KN | Saint Kitts and Nevis |
| LC | Saint Lucia |
| PM | Saint Pierre and Miquelon |
| VC | Saint Vincent and the Grenadines |
| WS | Samoa |
| SM | San Marino |
| ST | Sao Tome and Principe |
| SA | Saudi Arabia |
| SN | Senegal |
| RS | Serbia |
| SC | Seychelles |
| SL | Sierra Leone |
| SG | Singapore |
| SK | Slovakia |
| SI | Slovenia |
| SB | Solomon Islands |
| SO | Somalia |
| ZA | South Africa |
| GS | South Georgia and the South Sandwich Islands |
| ES | Spain |
| LK | Sri Lanka |
| SD | Sudan |
| SR | Suriname |

| Code | Country |
|---|---|
| SJ | Svalbard and Jan Mayen |
| SZ | Swaziland |
| SE | Sweden |
| CH | Switzerland |
| SY | Syrian Arab Republic |
| TW | Taiwan |
| TJ | Tajikistan |
| TZ | Tanzania, United Republic of |
| TH | Thailand |
| TG | Togo |
| TK | Tokelau |
| TO | Tonga |
| TT | Trinidad and Tobago |
| TN | Tunisia |
| TR | Turkey |
| TM | Turkmenistan |
| TC | Turks and Caicos Islands |
| TV | Tuvalu |
| UG | Uganda |
| UA | Ukraine |
| AE | United Arab Emirates |
| GB | United Kingdom |
| US | United States |
| UM | United States Minor Outlying Islands |
| UY | Uruguay |
| UZ | Uzbekistan |
| VU | Vanuatu |
| VE | Venezuela |
| VN | Vietnam |
| VG | Virgin Islands, British |
| VI | Virgin Islands, U.S. |

| Code | Country |
|---|---|
| WF | Wallis and Futuna |
| EH | Western Sahara |
| YE | Yemen |
| ZM | Zambia |
| ZW | Zimbabwe |

# Appendix B

## Flash Content Security Scenarios

This section provides more detailed scenarios on how to secure a live event and on-demand content on the Flash Media Streaming platform.

### Flash Media Streaming (Live StreamCast)

This section illustrates how Token-Based Authentication and the encoder settings used to publish your stream determine whether a live stream will be accessible.

Encoder configuration #1:

- **Publishing Point URL:** rtmp://fso.lax.0001.edgecastcdn.net/200001/Videos
- **Stream:** MyStream

Encoder configuration #2:

- **Publishing Point URL:** rtmp://fso.lax.0001.edgecastcdn.net/200001/Secure
- **Stream:** MyStream

Encoder configuration #3:

- **Publishing Point URL:** rtmp://fso.lax.0001.edgecastcdn.net/200001/Secure/Videos/2012
- **Stream:** MyStream?c1019f8a6942b46a1ce679e66cd579767

We will now examine the effect of securing a location called "/Secure" will have on the above encoder configurations.

**Reminder:** A location can be secured under the **Directories to Authenticate** section on the **Token Auth** page corresponding to the desired platform (e.g., Flash Media Streaming).

1. The first encoder configuration points to a live stream whose relative path is "/Videos." This asset does not require a token since the specified location is not protected by Token-Based Authentication. As a result, this asset will be served to the client.
2. The second encoder configuration points to a live stream whose relative path is "/Secure." This asset requires a token since it is stored in a folder protected by Token-Based Authentication. This asset will not be served to the client since a token was not specified for this request.

3. The third encoder configuration points to a live stream that uses a publishing point URL path that points to a subfolder of a protected folder. As a result, this stream is also protected by Token-Based Authentication. The requested asset will be delivered to the client, as long as the token is valid and the user requesting it meets the requirements specified in the provided token.

We have just examined how three encoder configurations would be affected when the "/Secure" location was secured for the Flash Media Streaming platform. We will now examine how alternate configurations will affect live events streamed for those same publishing point URLs.

**Note:** Each row in the following table represents a separate Token-Based Authentication configuration.

| Secured Location | Description |
|---|---|
| / | Live streams generated by all three encoder configurations will require a valid token. |
| /Secure/Videos | Live streams generated by the third encoder configuration will require a valid token. |
| /Secure/Videos/2012/MyStream | Live streams generated by all three encoder configurations will not be protected by Token-Based Authentication. |
| /200001 | Live streams generated by all three encoder configurations will not be protected by Token-Based Authentication. |

## Flash Media Streaming (On-Demand Content)

This section illustrates how the following URLs interact with Token-Based Authentication.

1. rtmp://fms.0001.edgecastcdn.net/000001/Secure/Presentation01.flv
2. rtmp://fms.0001.edgecastcdn.net/000001/Secure/2012/Presentation01.flv?c1019f8a6942b46a1ce679e66cd579767
3. rtmp://fms.0001.edgecastcdn.net/800001/secure.mydomain.com/Secure/Presentation01.flv
4. rtmp://secure2.mydomain.com/Presentation01.flv?c1019f8a6942b46a1ce679e66cd579767

We will now examine the effect of securing a location called "/Secure" will have on the above URLs.

1. The first URL points to an asset stored in "Secure" on a CDN origin server. This asset requires a token since it is stored in a folder protected by Token-Based Authentication. The asset will not be served to the client since a token was not specified for this request.

2. The second URL points to an asset stored on a CDN origin server. Since this asset is located in a subfolder of a protected folder, it is also protected by Token-Based Authentication. The requested asset will be delivered to the client, as long as the token is valid and the user requesting it meets the requirements specified in the provided token.
3. The third URL points to a customer origin server. The "secure.mydomain.com" folder is the name assigned to the customer origin configuration for the server hosting your assets. Although the requested asset is in a folder called "Secure," it will not be secured by Token-Based Authentication since the customer origin name was not included in the specified location. As a result, this request will be served to the client.
4. The fourth URL uses an edge CNAME in the URL. In this particular case, "secure2.mydomain.com" points to a folder called "Secure" that is located on a customer origin server called "secure.mydomain.com." Although the requested asset is in a folder called "Secure," it will not be secured by Token-Based Authentication since the customer origin name was not included in the specified location. As a result, this request will be served to the client.

We have just examined how different URLs would be affected when the "/Secure" location was configured for Token-Based Authentication for the Flash Media Streaming platform. We will now examine how alternate configurations will affect on-demand content streamed for those same URLs.

**Note:** Each row in the following table represents a separate Token-Based Authentication configuration.

| Secured Location | Description |
|---|---|
| / | All requests will require a valid token. |
| /Secure/2012 | Only the second URL will require a valid token. |
| /Secure/2012/Presentation01 | None of the requested assets will be protected by Token-Based Authentication. |
| /secure.mydomain.com | The third and fourth URLs will require a valid token. |
| /secure.mydomain.com/Secure | The third and fourth URLs will require a valid token. |

# 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

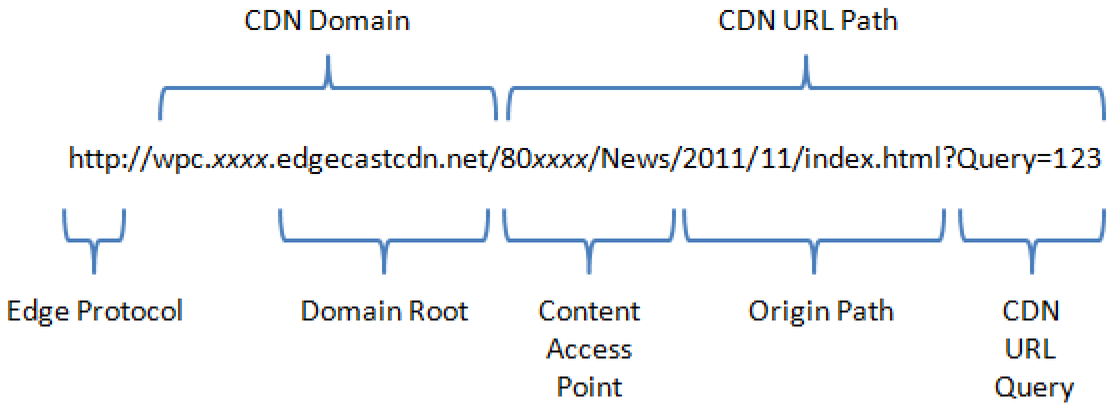
This term refers to a storage server on our CDN. Our CDN origin servers are in close proximity to our POPs, in order to provide optimal conditions for transferring data from a CDN origin server to your end-users via our POPs.

### 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.

- /00*xxxx*
- /00*xxxx*/Videos/2012/11/
- /00*xxxx*/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 is protecting the requested content, then 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 "*/yyxxxx/path*," where *yy* stands for the identifier and *xxxx* 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: "http://wpc.*xxxx*.edgecastcdn.net/00*xxxx."* In the following two examples, the edge CNAME references the following CDN URL: "http://wpc.*xxxx*.edgecastcdn.net/00*xxxx*/Videos."

| Edge CNAME URL | Points To |
|---|---|
| http://www.MyDomain.com/ | http://wpc.*xxxx*.edgecastcdn.net/00*xxxx*/ |
| http://www.MyDomain.com/2012/11/ | http://wpc.*xxxx*.edgecastcdn.net/00*xxxx*/Videos/2012/11/ |
| http://www.MyDomain.com/2012/11/Presentation01.flv | http://wpc.*xxxx*.edgecastcdn.net/00*xxxx*/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: "http://wpc.*xxxx*.edgecastcdn.net/00*xxxx*/Videos."

| Edge CNAME URL Path | Actual Edge CNAME URL |
|---|---|
| /2012/11/ | http://wpc.*xxxx*.edgecastcdn.net/00*xxxx*/Videos/2012/11/ |
| /2012/11/Show01.flv | http://wpc.*xxxx*.edgecastcdn.net/00*xxxx*/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. The token delimiter that you should use depends on whether you are using Live Streaming or Live StreamCast.

- Notation for Live StreamCast: <*StreamName*>?<*LiveAuthenticationKey*>
- Notation for Live Streaming: <*StreamName*>/<*LiveAuthenticationKey*>

### 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 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 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). This information can be used in a variety of ways. Our CDN allows you to leverage this information to determine how content will be cached. Additionally, you can choose to store query string information 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.