

Edgecast

Token-Based Authentication Administration Guide

edgecast

Disclaimer

Care was taken in the creation of this guide. However, Edgecast 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 Inc.

WINDOWS is a registered trademark of Microsoft Corporation.

About This Guide

Token-Based Authentication Administration Guide

Version 2.51

11/22/2021

© 2021 Edgecast Inc. All rights reserved.

Table of Contents

Introduction	1
Overview	1
Basic Setup and Usage	1
Phase 1: Platform-Specific Configuration	2
Phase 2: Content Linking.....	2
Request Handling.....	3
Platforms.....	4
Setting Up Token-Based Authentication.....	5
Overview	5
Setting an Encryption Key	5
Best Practices	6
Changing the Encryption Key	7
Setting Up Authentication by Folder	8
Sample Scenarios	11
Authentication Directory Administration	13
Setting Up Authentication by Request Type.....	14
Interaction with CDN Settings.....	14
Defining Requirements	15
Overview	15
Setting an Expiration Date	15
Restricting Access by Country.....	16
Restricting Access by URL	17
Restricting Access by Host	19
Restricting Access by Referrer	21
Restricting Access by IP address	23
Restricting Access by Protocol	23

Preventing Changes to Bandwidth Throttling Settings	24
Generating Tokens	25
Overview	25
Manually Generating a Token	26
Using Our Token Generator Application	26
Building a Custom Token Generator	28
Decrypting an Existing Token	28
Updating Linked Content	30
Overview	30
HTTP Large Example	30
HTTP Small Example	31
ADN Example	31
Redirecting Unauthorized Users	32
Quick Reference	33
Parameters	33
Bandwidth Throttling Reference	36
Appendix A	37
Country Codes (ISO 3166)	37

Introduction

Overview

The purpose of Token-Based Authentication is to provide a safeguard against hotlinking with the purpose of ensuring accurate billing of content delivery. It achieves this goal by requiring requests to contain a token value that defines the criteria that a requester must meet before it can be served via the CDN. This criteria may identify authorized requesters by:

- Country
- URL
- Host
- Referrer
- IP address
- Protocol

In addition to the above criteria, an expiration date may also be assigned to a token to ensure that a link only remains valid for a limited time period.

Basic Setup and Usage

Token-Based Authentication setup consists of defining a platform-specific configuration and updating links. This type of setup will cause the CDN to require authentication prior to content delivery. This basic workflow is outlined below.

Workflow	Description
Setup	<p>Platform-specific Configuration</p> <ol style="list-style-type: none">1. Define an encryption key.2. Apply security by:<ul style="list-style-type: none">• Directory• Request type <p>Content Linking</p> <ol style="list-style-type: none">1. Generate tokens.2. Update links (href and src).

Workflow	Description
CDN Traffic	Requests for secured content: <ol style="list-style-type: none">1. Require a valid token.2. Must meet the requirements defined in the token.

Phase 1: Platform-Specific Configuration

Perform the following steps to define the content that will require authentication:

1. Decide which platforms will require authentication.
2. Define an encryption key for each desired platform.
3. Define the set of content that will require authentication. This may be defined by either:
 - **Directory:** Define a directory whose contents will require authentication. Token-Based Authentication will be applied to all requests for content in that directory or a sub-folder of that location.
 - **HTTP Rules Engine:** A rule can be created within HTTP Rules Engine, which must be purchased separately, that enables or disables Token-Based Authentication when a request meets predefined criteria.

Phase 2: Content Linking

The next phase involves the following steps:

1. Generating encrypted tokens that define the minimum access requirements.
2. Updating CDN/edge CNAME URLs defined in href and src attributes to include the above encrypted token value as a query string parameter.

Note: Only the content defined in phase 1 will require authentication. All other content may be accessed using a standard CDN or edge CNAME URL.

Sample request:

[http://cdn.mydomain.com/secure/product.pdf?**1234567890abcdefgh**](http://cdn.mydomain.com/secure/product.pdf?1234567890abcdefgh)

The above request's query string, which is marked in bold, blue font, represents a token value.

Request Handling

A request for content defined in phase 1 must meet the following criteria:

- It must contain a valid token.
- It must satisfy the requirements defined in the token.

More Information:

1. An authorized request must contain a valid token that is appended to the file name in the CDN or edge CNAME URL.
 - **Sample Request:**
`http://data.server.com/asset.txt?c1019f8a6942b46a1ce679e168d5797670f3ee7e39068054ee4534d8a5a859d`
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 requirements for the requested content.
3. The user must satisfy all of the requirements defined for the requested content.
 - Content will be delivered when the requester meets all of the requirements defined in the decrypted token value.
 - The request will be denied when the requester cannot satisfy one or more requirements. Denied requests may be redirected to a different web page.

Platforms

Token-Based Authentication can require authentication for content delivered through any of our platforms (e.g., HTTP Large or HTTP Small). However, each platform must be configured individually. This allows additional flexibility when setting up Token-Based Authentication. The following table lists the available features and parameters.

Feature/Parameter
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.

Setting Up Token-Based Authentication

Overview

A valid Token-Based Authentication setup requires that the following steps be performed:

1. Set an encryption key.
2. Define the set of content that will require authentication by performing one or both of the following actions:
 - Identify one or more location(s) whose content will require a valid token value.
 - Identify the type of requests that will require a token value.
3. Update links to include a valid token value.

Setting an Encryption Key

An encryption key must be defined for each platform on which Token-Based Authentication will be applied. This platform-specific encryption key is used to generate and decrypt token values.

Key information:

- An encryption key may consist of any combination of alphanumeric characters. All other characters, including spaces, are not valid for encryption keys.
- An encryption key is case-sensitive. In other words, the case of an encryption key affects the encryption and decryption of token values.
- The maximum length of an encryption key is 250 characters.
- A minimum encryption version must be assigned to each key.
 - **V2:** Indicates that the key may be used to generate both version 2.0 and 3.0 tokens. This option should only be used when transitioning from a legacy version 2.0 encryption key to version 3.0.
 - **V3 (Recommended):** Indicates that the key may only be used to generate version 3.0 tokens.
- It may take up to an hour for changes, such as setting an encryption key or adding directory authentication), to take effect.

- By default, a token value is only specific to an encryption key and not to a folder or a platform. This means that it may be possible for a client to use a single token value to gain access to protected content from various folders across different platforms.
 - **Cross-Platform Access:** Assign a unique encryption key to each platform to prevent a single token from being used across multiple platforms.
 - **Cross-Folder Access:** Leverage the Allow URL parameter to ensure that a token may only be used for a specific directory or for a particular file.

To set an encryption key

1. Navigate to the **Token Auth** page corresponding to the desired platform.
2. Set the desired encryption key in the **Primary Key** option.
3. Make sure that the primary key's **Minimum Encryption Version** option is set to "V3."
4. Click **Update**.

Best Practices

Ensure token security by following these guidelines when defining an encryption key:

- Set it to a random value.
- Make sure that the encryption key candidate meets or exceeds the recommended length (i.e., 64 characters).

Reminder: Do not exceed a key length size of 250 characters.

OpenSSL

A standard method for generating random values is to use the OpenSSL tool to perform hexadecimal encoding.

Syntax:

```
rand -hex KeyLength
```

Hexadecimal encoding doubles the specified length. For example, specifying a length of "32" will generate a value containing 64 characters.

Example:

```
OpenSSL> rand -hex 32
Loading 'screen' into random state – done
70ae02ac9f8270e160eadbaefdd5df37c8e13750d1793dcd55b00943fff3b829
```

Changing the 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 the requested content will not be delivered. 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 a new encryption key to take effect (approximately 1 hour).

As a result of all of these factors, it is recommended to leverage two active encryption keys to ensure that authorized users enjoy uninterrupted access to your content. This procedure requires that the old key be assigned as a backup key when creating a new encryption key. Since the old key is still an active encryption key, users will still be able to access your content using old tokens. The old encryption key may be safely removed after all of the following events have taken place:

- The new encryption key has taken effect.
- All of your links have been updated.
- Cached content that leveraged an old token is no longer being served.

The above process ensures a smooth transition to a new encryption key.

To change your encryption key (recommended procedure)

1. Navigate to the **Token Auth** page corresponding to 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. Make sure that the primary key's **Minimum Encryption Version** option is set to "V3."
5. Click **Update** to save your changes. It may take up to an hour for your primary key to become active.
6. Generate new tokens using the new primary key.
7. Update all links to content secured by Token-Based Authentication to use the tokens generated in the previous step.
8. Purge the content updated in the previous step.

9. 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.

Setting Up Authentication by Folder

Before discussing how to define directory authentication, let us review the following points:

- Token-Based Authentication must be configured on each desired platform (e.g., HTTP Large or HTTP Small).
 - A primary and/or backup encryption key must be specified for each desired platform.
- Defining an encryption key by itself will not affect content delivery.

The set of content that requires authentication needs to be identified. One way of accomplishing this is to specify the location(s) for which Token-Based Authentication will be applied. The **Directories to Authenticate** section allows you to define one or more locations using a relative path to the desired folder. The starting point for this relative path, which varies by URL type, is defined below:

URL Type	Relative Path (Starting Point)
CDN URL	<p>Specify a relative path that starts directly after the account number segment of the content access point (e.g., /000001, /200001, or /800001).</p> <p>Sample URL:</p> <p>http://wac.0001.edgecastcdn.net/800001/customerorigin/videos/fly.flv</p> <p>In the above sample URL, the gray text indicates what should be excluded when securing a location. This sample request can be secured by any of the following configurations:</p> <ul style="list-style-type: none"> • / • /customerorigin • /customerorigin/videos

URL Type	Relative Path (Starting Point)
Edge CNAME URL (CDN Origin)	<p>Specify a relative path that starts directly after the hostname.</p> <p>Sample URL:</p> <p><code>http://www.domain.com//presentations/sales/businessplan.ppt</code></p> <p>In the above sample URL, the gray text indicates what should be excluded when securing a location. This sample request can be secured by any of the following configurations:</p> <ul style="list-style-type: none"> • / • /presentations • /presentations/sales
Edge CNAME URL (Customer Origin)	<p>Specify a relative path that starts with the name of the customer origin configuration referenced by the edge CNAME URL.</p> <p>Sample edge CNAME URL:</p> <p><code>http://www.domain.com/Photos/Store.jpg</code></p> <p>Our edge servers will re-write the edge CNAME URL requested by the client (above) with the following CDN URL:</p> <p><code>http://wac.0001.edgecastcdn.net/800001/customerorigin/Photos/Store.jpg</code></p> <p>The above CDN URL is used to determine the starting point for the relative path that should be secured. The gray text in the above CDN URL indicates what should be excluded when securing a location. This sample request can be secured by any of the following configurations:</p> <ul style="list-style-type: none"> • / • /customerorigin • /customerorigin/Photos

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.

Origin Server

Require authentication for all CDN content by defining the following relative path under the **Directories to Authenticate** section:

- /

Require authentication for all content from a specific customer origin server through the following configuration:

- /CustomerOriginName

Require authentication for all content from a specific folder on a specific customer origin server through the following configuration:

- /CustomerOriginName/Path

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 defining such a location make sure to specify the name of the customer origin server followed by the relative path to the desired folder (e.g., /MyCustomerOrigin/Marketing/Presentations).

Note: There is an exception that only applies to the HTTP Large, HTTP Small, 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

Token-Based Authentication is applied recursively to each folder specified in the **Directories to Authenticate** section. This means that all content residing in the specified folder or its subfolders will require authentication.

Important: Token-Based Authentication is platform-specific which means that content may potentially be downloaded by using a URL for a different platform (e.g., HTTP Large instead of HTTP Small). Avoid this scenario by replicating your Token-Based Authentication configuration across all platforms.

Tip: Due to the recursive nature of directory authentication, apply Token-Based Authentication to an entire platform by adding the root folder (/).

To apply Token-Based Authentication across an entire platform

1. Navigate to the **Token Auth** page on the desired platform.
2. Set the **New** option, which can be found in the **Directories to Authenticate** section, to forward slash (/).
3. Click **Add**.

Sample Scenarios

This section illustrates how the following requests 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 platform, the analysis provided below also applies to the HTTP Small 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.

Authentication Directory Administration



The directories to which Token-Based Authentication will be applied can be administered on a per platform basis. Add, modify, or delete each directory from the **Token Auth** page corresponding to the desired platform.

Note: It may take up to an hour for the creation, modification, or deletion of an authentication directory to take effect.


To add an authentication directory

1. From the MCC, navigate to the **Token Auth** page for the desired platform.
2. In the **New** option, which can be found in the **Directories to Authenticate** section, type the relative path to the desired folder.
3. Click **Add**.

To modify an authentication directory

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 desired directory.
4. Click .

To delete an authentication directory

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.

Setting Up Authentication 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 detailed information on HTTP Rules Engine, please refer to the **HTTP Rules Engine Administration Guide**.

Note: 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 platform configuration and then use 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.

Defining Requirements

Overview

A token value may consist of a set of requirements that the requester must satisfy before the requested content is delivered. These requirements may be mixed and matched as desired. Adding, modifying, or removing requirements simply involves generating 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: Tokens can be specific to a particular folder or asset. Take advantage of this capability to prevent a token from being reused to access other protected content.

Note: Invalidate old token values by changing the primary and backup encryption keys.

As previously noted, a token value is required for the delivery of content for which Token-Based Authentication has been applied. This token value may either be the same for all content or the set of requirements may vary according to the content being requested.

Setting an Expiration Date

Time-sensitive content may be assigned a token that expires after a specified time period. Requests that are submitted after the expiration date will be denied.

The Expire parameter (i.e., `ec_expire`) defines an expiration date and time by the number of seconds that have elapsed since Unix time (a.k.a. POSIX time or Unix epoch). Unix time starts on 1970-01-01 at 00:00:00 GMT.

Tip: Language-specific functions that provide Unix time conversions are widely available. However, at times it may be necessary to manually set this parameter or to troubleshoot an existing token. There are a variety of websites that provide conversion to and from standard time conventions to Unix time.

Restricting Access by Country

The Allow Country (i.e., `ec_country_allow`) and the Deny Country (i.e., `ec_country_deny`) parameters may be used to restrict access by the country from which the request originated.

Restriction	Parameter	Description
Allow	<code>ec_country_allow</code>	This parameter only allows requests that originate from one or more specified countries. Requests that originate from all other countries will be denied access.
Deny	<code>ec_country_deny</code>	This parameter denies requests that originate from one or more specified countries. Requests that originate from all other countries will be allowed.

Note: An alternative method for restricting access by country is the Country Filtering feature. This feature, which is supported on the HTTP Large, HTTP Small, and the ADN platforms, is available from the Country Filtering page. The Country Filtering feature is applied to all platform-specific traffic and therefore doesn't require the use of tokens, which reduces setup time. For more information on country filtering options, please refer to the **HTTP Large**, **HTTP Small**, or the **ADN Administration Guide**.

Key information:

- These parameters may be set to any two-letter ISO 3166 country code. Valid country codes are listed in **Appendix A: Country Codes**.
- Specify multiple countries by separating each country code with a comma (as shown below).
 - `US,GB,MX,FR`
 - Do not add a space when delimiting countries. Country codes that are preceded by a space will be excluded from a token's requirements.
- Country codes are case-insensitive.
- In the atypical scenario where a token contains both the Allow Country and the Deny Country parameters, the Allow Country parameter (i.e., `ec_country_allow`) takes precedence over the Deny Country parameter (i.e., `ec_country_deny`).

Restricting Access by URL

The tokens generated from most parameters are not specific to a particular folder or asset. Therefore, those tokens may potentially be reused to authenticate content stored in various folders. The Allow URL parameter (i.e., `ec_url_allow`), on the other hand, tailors tokens to a particular asset or path. This parameter restricts access to requests whose URLs start with a specific relative path. The configuration for this parameter varies according to the platform through which your protected content will be accessed.

Allow URL Parameter & HTTP-Based Platforms

The Allow URL parameter checks whether the relative path to the requested content starts with the relative path specified by this parameter. This allows the scope of this parameter to be as broad or as specific as your needs require.

For example, if this parameter is set to `"/marketing,"` then it will be satisfied by any of the following requests:

- `http://cdn.mydomain.com/marketing.htm`
- `http://cdn.mydomain.com/marketing/images/presentation01.jpg`
- `http://cdn.mydomain.com/marketingmaterials/images/marketing.htm`

However, if this parameter is set to `"/marketing.htm,"` then it will only be satisfied by the first request.

The starting point for the comparison between the two relative paths (i.e., requested content vs. parameter value) is immediately after the domain portion of the URL. This starting point applies to both CDN and edge CNAME URLs.

The following table provides a variety of sample URLs. Bold font indicates the portion of the URL that will be compared against the relative path defined for this parameter.

Type	URL
CDN URL	<code>http://wac.0001.edgecastcdn.net/800001/MyServer/marketing.htm</code>
Edge CNAME URL	<code>http://cdn.mydomain.com/marketing.htm</code>

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.

Example

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 platform, the analysis of these URLs also applies to the HTTP Small 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`

Restricting Access 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, HTTP Small, 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 Generator application or by creating your own token generator.

The 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 parameter. This 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 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`

Restricting Access by Referrer

Requests may be allowed or blocked based on referrer. A referrer identifies the URL for the web page from which the link was followed. The two parameters that leverage referrer information are identified below.

Parameter	Description
ec_ref_allow	Only requests that match the specified referrer will be allowed.
ec_ref_deny	Only requests that match the specified referrer will be denied.

Key information:

- The referrer value defined in the parameter 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 hostname and/or a particular path on that hostname.
- Validate multiple referrers within a single parameter by separating each one with a comma.
- Although a typical configuration should not include both parameters, it is possible for a token to contain both of these parameters. In such a case, the ec_ref_allow parameter takes precedence over the ec_ref_deny parameter.

Important: 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.

Wildcard Matching for Subdomains (HTTP-Based Platforms)

The HTTP Large, HTTP Small, and the ADN platform support the use of 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 parameter on the HTTP Large, HTTP Small, 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 parameter on the HTTP Large, HTTP Small, or the ADN platforms.

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

- [Blank or not provided]
- `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`

Restricting Access 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).

Restricting Access by Protocol

You can choose to allow or block users depending on the protocol used to request the desired content. The 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: 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 Generator application or by creating your own token generator.

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

Generating Tokens

Overview

Important: You should upgrade to Token-Based Authentication 3.0. [Learn more \(CDN Help Center\)](#).

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 requirements. When generating a token, keep in mind that there is no limit to the number of 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 requirements that must be met prior to content delivery.

Important: Although there is no limit to the number of 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: Generating tokens will not affect your Token-Based Authentication configuration in any way. Additionally, there is no limit to the number of token values that may be generated for a particular encryption key.

We offer two direct ways to generate a token, which are through the MCC and using the Token Generator application (ectoken3). 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.

To manually generate a token

1. Review the available parameters.
2. Assign a value to each parameter that a client must meet before content may be delivered.
3. From the **Key to Encrypt** option, select the desired encryption key.
4. From the **Encryption Version** option, select the desired version. The recommended encryption version is 3.0.
5. Click **Encrypt** to generate a token specific to the selected key. This token value will appear next to the **Generated Token** label.
6. Modify the desired request to include the above token value as a query string.

Example:

```
http://cdn.mydomain.com/sales.pdf?1234567890abcdefghijklmnop
```

Using Our Token Generator Application

An alternative method for generating token values is to use the Token Generator application. This application provides the means to generate tokens using a script. The advantage of this approach is that it allows token values to be tailored to content.

The Token Generator application includes the following components:

- Windows executable
- Linux binaries

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

The proper syntax for specifying a single parameter is described below.

Version 3.0 (Recommended):

```
ectoken3 KeyName "parameter=value"
```

Version 2.0:

```
ectoken3 -2 KeyName "parameter=value"
```

Version 2.0 Only: A version 2.0 token will only be generated when the "-2" parameter is the first parameter. You should upgrade to Token-Based Authentication 3.0. [Learn more \(CDN Help Center\)](#).

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

```
ectoken3 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 12/31/2015 12:00:00 GMT.
- Only allows access to North American countries.
- Only allows referrers from "TrustedDomain.com."

Then you would use the following syntax:

```
ectoken3 MyKey  
"ec_expire=1451563200&ec_country_allow=US,CA,MX&ec_ref_allow=*.TrustedDomain.com"
```

The token value associated with this configuration would be:

```
1ea46ba396e88f03a9f6b6b968b32d2fd88858148f120a1bbca7882de68b8b14a9bde8bcd6c36bcd30e8bbb47d9997ab7260381b4c1ed99de5baf805ed54fd3609e8066e43a92a5b2c7839ba95080d3668ab9dd47d9275d8eb29b8ccf8f49515745f18a66c
```

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

```
http://secure.server.com/MyProtectedAsset.html?1ea46ba396e88f03a9f6b6b968b32d2fd88858148f120a1bbca7882de68b8b14a9bde8bcd6c36bcd30e8bbb47d9997ab7260381b4c1ed99de5baf805ed54fd3609e8066e43a92a5b2c7839ba95080d3668ab9dd47d9275d8eb29b8ccf8f49515745f18a66c
```

Tip: The **Token Auth** page provides an "Encrypt Tool" that can generate tokens. This tool will also display the corresponding call through which our Token Generator application (i.e., ectoken3) can generate the same token. This sample syntax will appear next to the **Token Generator Call** label.

Building a Custom Token Generator

Leverage our open-source repository to create a custom application to generate token values. This repository, which is hosted on GitHub, contains C, C++, C#, PHP, Perl, Java, and Python source code.

This repository is located at:

<https://github.com/EdgeCast/ectoken>

Decrypting an Existing Token

Important: You should upgrade to Token-Based Authentication 3.0. [Learn more \(CDN Help Center\)](#).

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 requirements. If you suspect that a particular client is having trouble viewing your protected content, you can decrypt his/her token to discover which requirement is not being met.

A token value can be decrypted using any of the following:

- Decrypt Tool (**Token Auth** page)
- ectoken3 decrypt *KeyName Token*
- Custom token generator
Leverage our open-source repository to create a custom application to decrypt token values. This repository, which is hosted on GitHub, contains C, C++, C#, PHP, Perl, Java, and Python source code.
This repository is located at:
<https://github.com/EdgeCast/ectoken>

To decrypt a token using the Decrypt tool

1. Navigate to the **Decrypt Tool** section of the **Token Auth** page.
2. In the **Token To Decrypt** option, paste the desired token value.
3. In the **Key to Decrypt** option, select the encryption key used to generate that token value.
4. Click **Decrypt**. The requirements for that token will appear next to the **Original Parameters** label.

Note: The Decrypt tool will not be able to decrypt tokens generated with an old encryption key. Use the Token Generator executable to decrypt these types of token values.

To decrypt a token using the Token Generator (ectoken3) executable

1. Download the Token Generator executable.
2. Extract the Windows or Linux version to a local drive.
3. Open the command prompt.
 - **Windows:** (START + R, CMD)
 - **Linux:** Open a terminal window.
4. Navigate to the directory where the Token Generator executable is located.
5. Issue the following command: `ectoken3 decrypt KeyName Token`
 - **KeyName:** Replace this term with the encryption key used to generate the token value.
 - **Token:** Replace this term with the token value that you would like to decrypt.

Updating Linked Content

Overview

Providing access to 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 Example

The sample code excerpt provided below demonstrates how to insert a token into a CDN and an edge CNAME URL on the HTTP Large platform.

CDN URL:

The following sample CDN URL contains a custom query string parameter called "user." Custom query string parameters can be appended to the URL through the use of an ampersand.

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

Edge CNAME URL:

```

```

HTTP Small Example

The sample code excerpt provided below demonstrates how to insert a token into a CDN and an edge CNAME URL on the HTTP Small platform.

CDN URL:

The following sample CDN URL contains a custom query string parameter called "user." Custom query string parameters can be appended to the URL through the use of an ampersand.

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

Edge CNAME URL:

```

```

ADN Example

The sample code excerpt provided below demonstrates how to insert a token into 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.

CDN URL:

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

Edge CNAME URL:

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

Redirecting Unauthorized Users

By default, a user that does not meet the minimum authentication requirements defined for the requested content will view a web page with a 403 Forbidden status code. If the protected content is accessed through either the HTTP Large, HTTP Small, 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

Parameters

This section provides a brief description for each available 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	<p>Restricts content delivery to requests that originate from a specific IP address. This parameter uses standard IPv4 notation.</p> <p>Sample value: 111.11.111.11</p> <p>This example will only serve an asset to a client with an IP address of 111.11.111.11.</p>
ec_country_allow	<p>Restricts content delivery to the specified countries. 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.</p> <p>Sample value: US</p> <p>This example will deny all requests that do not originate from the United States.</p>
ec_country_deny	<p>Blocks 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.</p> <p>Sample value: US,CA</p> <p>This example will deny all requests that originate from the United States and Canada.</p>
ec_expire	<p>Defines 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.</p> <p>Sample value: 1356955200</p> <p>This example will set the expiration date and time to 12/31/2012 12:00:00 GMT.</p>

Parameter	Description
ec_host_allow	<p>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.</p> <p>Sample value: server1.com,*.server2.com</p> <p>This example will allow access to the following hosts:</p> <ul style="list-style-type: none"> • server1.com • secure.server2.com
ec_host_deny	<p>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.</p> <p>Sample value: server1.com,*.server2.com</p> <p>This example will deny access to the following hosts:</p> <ul style="list-style-type: none"> • server1.com • secure.server2.com
ec_proto_allow	<p>Defines the protocol that can be used to retrieve an asset. Acceptable values for this parameter are "http" and "https."</p> <p>Sample value: https</p> <p>This example will only allow access to URLs that use the https protocol.</p>
ec_proto_deny	<p>Defines the protocol that cannot be used to retrieve an asset. Acceptable values for this parameter are "http" and "https."</p> <p>Sample value: http</p> <p>This example will deny access to URLs that use the http protocol.</p>

Parameter	Description
ec_ref_allow	<p>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.</p> <p>Sample value: server1.com/obj1,*.server2.com</p> <p>This example will allow access to the following referrers:</p> <ul style="list-style-type: none"> • server1.com/obj1/index.htm • server1.com/obj1.html • secure.server2.com/2012/Graph.xml
ec_ref_deny	<p>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.</p> <p>Sample value: server1.com/obj1,*.server2.com</p> <p>This example will deny access to the following referrers:</p> <ul style="list-style-type: none"> • server1.com/obj1/index.htm • server1.com/obj1.html • secure.server2.com/2012/Graph.xml
ec_url_allow	<p>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).</p> <p>Sample value: /000001/dir1/movie1,/000001/dir2</p> <p>Assuming that the above value was configured for the HTTP Large platform, this example will allow access for the following requests:</p> <ul style="list-style-type: none"> • http:// wpc.xxxx.edgecastcdn.net/00xxxx/dir1/movie1.flv • http:// wpc.xxxx.edgecastcdn.net/00xxxx/dir1/movie1.mpg • http:// wpc.xxxx.edgecastcdn.net/00xxxx/dir1/movie1/index.htm • http:// wpc.xxxx.edgecastcdn.net/00xxxx/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 Administration Guide**.

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

Name	Description
ec_prebuf	<p>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.</p> <p>Sample value: 10</p> <p>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.</p>
ec_rate	<p>This parameter limits the rate (KBps) at which clients will be able to download the specified asset.</p> <p>Sample value: 64</p> <p>This example will limit a user's download speed to 64 KBps.</p>

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** 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