# Amazon CloudFront

## Developer Guide

## API Version 2012-03-15

# Amazon CloudFront: Developer Guide

# Welcome

The *Amazon CloudFront Developer Guide* provides developers with a conceptual and architectural overview of Amazon CloudFront, plus it provides how-to information on working with objects and distributions, securing access to content, and using the CloudFront REST API.

*CloudFront* is a web service for content delivery. It integrates with other Amazon Web Services to give you an easy way to distribute content to end users with low latency and high data transfer speeds.

## How Do I...?

| How Do I? | Relevant Topics |
| --- | --- |
| Get Started | Amazon CloudFront Getting Started Guide |
| Learn if CloudFront is right for my use case | Introduction to Amazon CloudFront (p. 2) |
| Use CloudFront distributions | Working with Distributions (p. 35) |
| Migrate from Amazon S3 to CloudFront | Migrating from Amazon S3 to CloudFront (p. 19) |
| Create a streaming distribution | Working with Distributions (p. 35) |
| Understand the security protocols used with CloudFront | Creating Secure HTTPS Connections (p. 113) |
| Use the CloudFront API | Making API Requests (p. 129) |

# Introduction to Amazon CloudFront

**Topics**

This introduction to CloudFront is intended to give you a detailed summary of this web service. After reading this section, you should have a good idea of what CloudFront offers and how it can fit in with your business.

# What Is Amazon CloudFront?

CloudFront is a web service for content delivery. It makes it easier for you to distribute content to end users quickly, with low latency and high data transfer speeds.

CloudFront delivers your content through a worldwide network of edge locations. End users are routed to the nearest *edge location*, so content is delivered with the best possible performance. CloudFront works seamlessly with the Amazon Simple Storage Service, which durably stores the original, definitive versions of your files.

# CloudFront Concepts

This section discusses the basic concepts that you need to understand before using CloudFront.

## Objects

Objects are the files you want CloudFront to deliver. This typically includes web pages, images, and digital media files, but can be anything that can be served over HTTP or a version of RTMP.

## Origin Server

An origin server is the location where you store the original, definitive version of your objects. For CloudFront, your origin server is either an Amazon S3 bucket that you own, or a custom origin. For more information about origin servers, see The Origin Server (p. 38).

## Distributions

After you store your objects in your origin server, how do you get CloudFront to recognize them? You create a *distribution*, which is a link between your *origin server* and a domain name that CloudFront automatically assigns. If your origin is an Amazon S3 bucket, you use this new domain name in place of standard Amazon S3 references, for example, `http://myawsbucket.s3.amazonaws.com/image.jpg` would instead be `http://somedomainname.cloudfront.net/image.jpg`.

Distributions can incorporate CNAME aliases you want to use. For more information, see Using CNAMEs (p. 49).

A *streaming distribution* is like a distribution except that a streaming distribution delivers digital media over a Real-Time Messaging Protocol (RTMP) connection (instead of HTTP). Adobe's Flash Media Server can serve RTMP and the streaming distribution can be used as it downloads. The objects stored in Amazon S3 are different according to the distribution type.

## Edge Locations

An *edge location* is a geographical site where CloudFront caches copies of your objects. When an end user requests one of your objects, CloudFront decides which edge location is best able to serve the request. If the edge location doesn't have a copy, CloudFront goes to the origin server and puts a copy of the object in the edge location.

For the locations of CloudFront edge locations, go to the CloudFront Details page.

## Expiration

By default, an object expires after being in an edge location for 24 hours. After the object expires, CloudFront no longer serves that particular object and must get a new copy of that object from the origin server to serve to end users. The minimum expiration time is 0 seconds; there isn't a maximum expiration time limit. For more information about expiration, see Specifying When an Object Expires (p. 25).

## Eventual Consistency

When you successfully create, modify, or delete a distribution using the CloudFront *control API*, it takes time for your changes to propagate throughout the CloudFront system. This information about the distribution is eventually consistent, but an immediate request to the control API to get that distribution's information might not show the change. Consistency is usually reached within minutes, but a high system

load or network partition might increase this time. The control API lets you determine when your changes have been fully deployed.

# Architectural Overview

These are the main actors involved when you use CloudFront:

- You
- Your website or application
- Your origin server (usually Amazon S3)
- CloudFront
- End users using your website or application

Using CloudFront involves the types of communication described in the following figure and table.
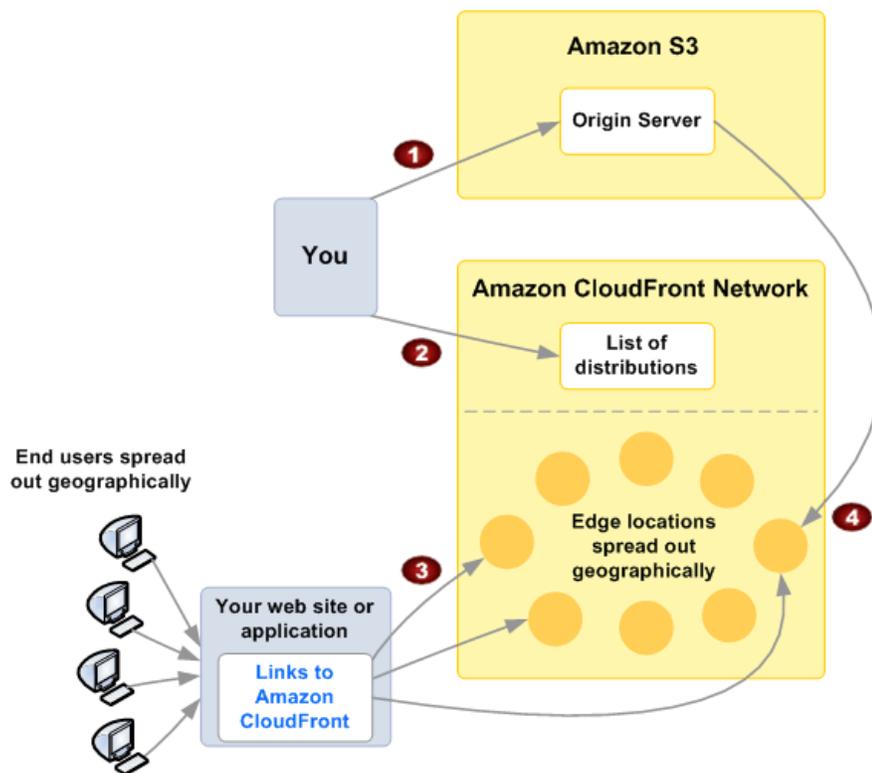
| | Communication Between | Reason |
|---|---|---|
| ❶ | You and Your Origin | To put objects in the origin server, manage them, and delete them. |
| | | If using Amazon S3, this is the normal interaction you have with Amazon S3 to manage buckets and objects. This relationship is covered in the Amazon Simple Storage Service Developer Guide. |
| ❷ | You and CloudFront | To manage your *distribution*. |
| | | This communication tells CloudFront the location of your origin server and lets you configure aspects of how CloudFront should deliver your content to end users. You can communicate with CloudFront through the REST *control API*. This guide covers how to do this (for more information, see Making API Requests (p. 129), and go to Amazon CloudFront API Reference). |
| ❸ | End users and CloudFront | To get copies of your objects from the edge locations (via your website or application). |
| | | This involves how end users link to your objects in CloudFront through your website or application. Because CloudFront (and not Amazon S3) serves the objects, you give end users links to CloudFront (instead of links to the objects in Amazon S3). For more information about the format of these links, see Format of URLs for CloudFront Objects (p. 21). |
| ❹ | CloudFront and Your Origin Server | To get copies of your objects from the origin server as needed. |
| | | CloudFront takes care of this communication automatically when a request comes in to an edge location that doesn't already have a copy of the object. |

# How CloudFront Delivers Content

The following figure and table describe the basic process CloudFront uses to deliver your content. In this example, the content is a file called `image.jpg`, and the content origin is an Amazon S3 bucket.

> ⚠️ **Important**
>
> The following process assumes that you make the objects in your bucket publicly readable (which means anyone who knows the bucket's name and object's name could access the object). If you'd prefer to keep the objects private and control who accesses them, see Using a Signed URL to Serve Private Content (p. 70).
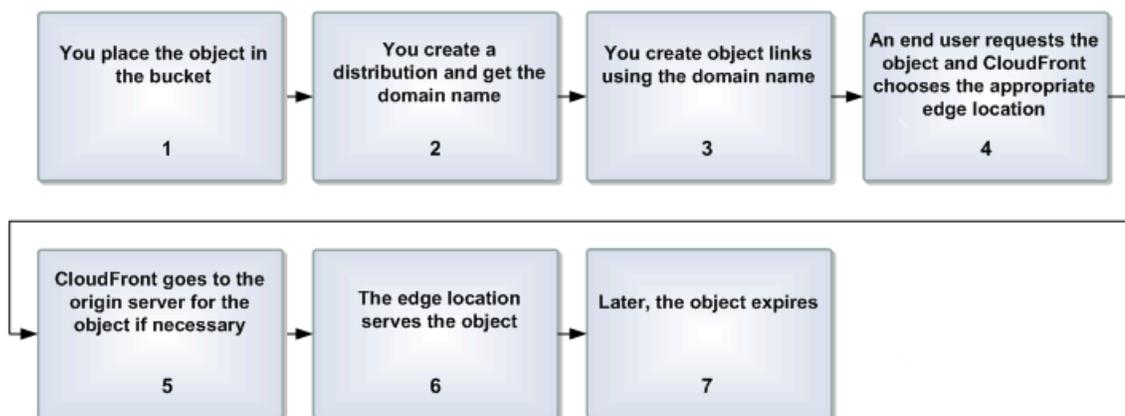


**Process for Delivering Content**

| 1 | You place the original version of `image.jpg` in your Amazon S3 origin server bucket and make it publicly readable. |
|---|---|
| 2 | You create a *distribution* and get your distribution's domain name. |
| 3 | You create links to `image.jpg` in your website or web application with the CloudFront domain name. |
| 4 | When an end user requests a page that contains `image.jpg`, CloudFront determines which edge location would be best to serve `image.jpg` (in this case, we'll say it's the St. Louis location). |
| 5 | If the St. Louis edge location doesn't have a copy of `image.jpg`, CloudFront goes to the origin server and puts a copy of `image.jpg` in the St. Louis location. |
| 6 | The St. Louis edge location then serves `image.jpg` to the end user and then serves any other requests for that file at the St. Louis location. |
| 7 | Later, `image.jpg` expires, and CloudFront deletes `image.jpg` from the St. Louis location. CloudFront doesn't put a new copy of `image.jpg` in the St. Louis location until an end user requests `image.jpg` again and CloudFront determines the St. Louis location should serve the image. |

CloudFront repeats tasks 4–7 as needed to satisfy end-user demand for `image.jpg`.

**Related Topics**

# Locations of CloudFront Edge Servers

For a list of the locations of CloudFront edge servers, see The Amazon CloudFront Edge Network on the Amazon CloudFront detail page.

# Paying for CloudFront

CloudFront is designed so you don't have to pay any upfront fees or commit to how much content you'll have. Like with the other AWS services, you pay as you go, and only for what you use.

The following diagram and table summarize the charges to use CloudFront.
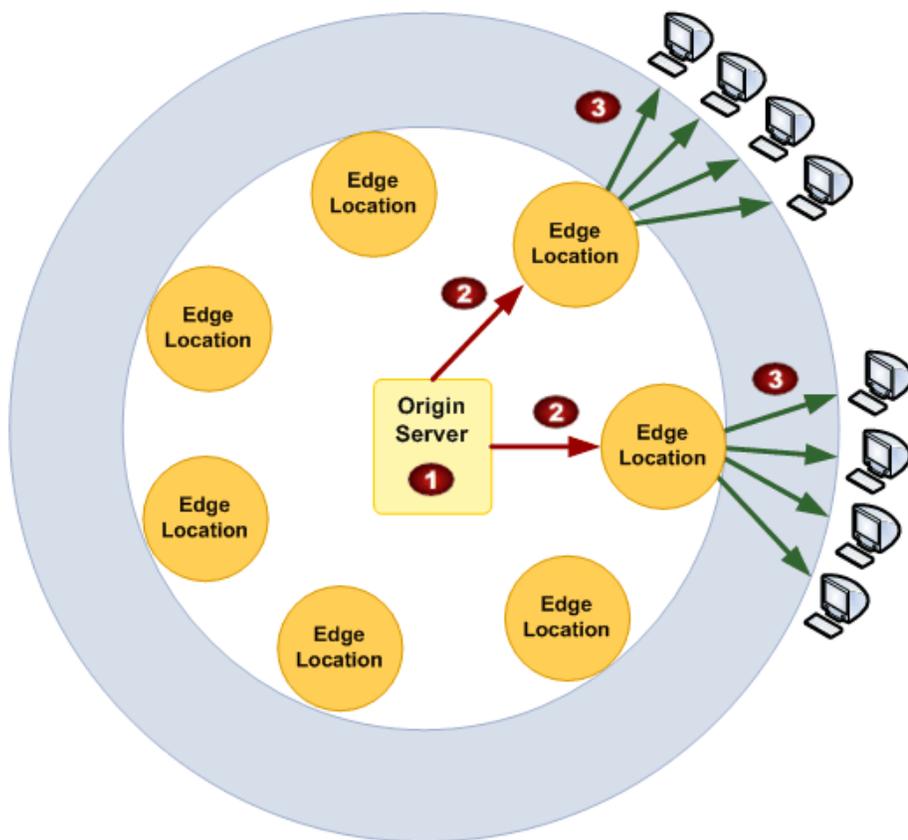


Your monthly bill from AWS separates your usage and dollar amounts by AWS service and function. As a result, you see some charges for storing objects with Amazon S3 (1) (if you are using Amazon S3 as your origin server), some charges for data transfer between your bucket and your edge location (2), and some charges for serving data from CloudFront (3).

|   | Charge | Comments |
|---|--------|----------|
| **1** | Storage in an Amazon S3 origin server | You pay normal Amazon S3 storage charges to store objects in your bucket; the charges appear in the Amazon S3 portion of your AWS statement. |

| | Charge | Comments |
|---|---|---|
| ❷ | Copying objects to edge locations | If using an Amazon S3 origin server, you incur the normal Amazon S3 charges for GET requests and for data transfer out. CloudFront copies an object to an edge location only if there is demand for that object at that edge location. |
| | | The data transfer charges appear in the AWS Data Transfer portion of your AWS statement. |
| ❸ | Serving objects from edge locations | You incur CloudFront charges for requests and data transfer out, which are lower than the corresponding Amazon S3 charges. The CloudFront charges appear in the CloudFront portion of your AWS statement. |

# CloudFront or Amazon S3?

Both CloudFront and Amazon S3 serve content. Should you use CloudFront to serve all your content? Not necessarily. It depends on your particular needs.

Amazon S3 is designed to store the original, definitive version of your files. It is optimized for high durability and cost-effective application storage and data transfer.

CloudFront is designed to distribute your most popular content with low latency. It is not designed for durable storage. Copies of your popular objects are stored in edge locations close to end users on the Internet; if an object isn't accessed frequently it might be removed from an edge location. For objects that are served many times, CloudFront can lower the cost of delivery while providing a faster download experience.

If you expect a large number of requests for each of your files, CloudFront provides higher performance than Amazon S3 alone, because objects are stored closer to end users' locations. You might also find CloudFront a more cost-effective choice than Amazon S3 for delivery of popular objects due to its lower charges for data transfer at higher usage tiers.

# Controlling User Access to Your AWS Account

**Topics**

Amazon CloudFront integrates with AWS Identity and Access Management (IAM) so that you can create Users for your AWS Account and you can specify which CloudFront actions a User (or a group of Users) can perform in your AWS Account. You control User access to CloudFront by creating policies that describe User or group permissions. For example, you might create a policy that gives only certain Users in your organization permission to use `GetDistributionConfig`. They could then use the action to retrieve data about your CloudFront distributions.

For more information on using policies to set AWS Account User permissions, go to Permissions and Policies in *Using AWS Identity and Access Management*. For general information about IAM, go to AWS Identity and Access Management on the AWS website.

> ⚠️ **Important**
>
> Using Amazon CloudFront with IAM doesn't change how you use CloudFront. There are no changes to CloudFront actions, and no new CloudFront actions related to Users and access control.

## CloudFront Resources

You use an asterisk (*) as the resource when writing a policy to control access to CloudFront actions. This is because you can't use IAM to control access to specific CloudFront resources. For example, you can't give Users access to a specific distribution. Permissions granted using IAM include all the resources you use with CloudFront. Because you cannot specify the resources to control access to, there are no CloudFront resource ARNs (Amazon Resource Names) for you to use in an IAM policy. (For detailed information about using ARNs with IAM, go to "ARNs" in the Identifiers for IAM Entities section of *Using AWS Identity and Access Management*.)

## CloudFront Actions

In an IAM policy, you can specify any and all API actions that CloudFront offers. The action name must be prefixed with the lowercase string `cloudfront:`. For example: `cloudfront:GetDistributionConfig`, `cloudfront:ListInvalidations`, or `cloudfront:*` (for all CloudFront actions).

The following tables list the canonical names for all CloudFront actions. Use these canonical names when specifying APIs in IAM policies.

| API Actions for Download Distributions | Canonical Name |
| --- | --- |
| POST Distribution | CreateDistribution |
| GET Distribution | GetDistribution |
| GET Distribution Config | GetDistributionConfig |

| API Actions for Download Distributions | Canonical Name |
|---|---|
| PUT Distribution Config | UpdateDistribution |
| GET Distribution List | ListDistributions |
| DELETE Distribution | DeleteDistribution |

| API Actions for Streaming Distributions | Canonical Name |
|---|---|
| POST Streaming Distribution | CreateStreamingDistribution |
| GET Streaming Distribution | GetStreamingDistribution |
| GET Streaming Distribution Config | GetStreamingDistributionConfig |
| PUT Streaming Distribution Config | UpdateStreamingDistribution |
| GET Streaming Distribution List | ListStreamingDistributions |
| DELETE Streaming Distribution | DeleteStreamingDistribution |

| API Actions for Invalidations | Canonical Name |
|---|---|
| POST Invalidation | CreateInvalidation |
| GET Invalidation | GetInvalidation |
| GET Invalidation List | ListInvalidations |

| API Action for Origin Access Identities | Canonical Name |
|---|---|
| POST Origin Access Identity | CreateCloudFrontOriginAccessIdentity |
| GET Origin Access Identity | GetCloudFrontOriginAccessIdentity |
| GET Origin Access Identity Config | GetCloudFrontOriginAccessIdentityConfig |
| PUT Origin Access Identity Config | UpdateCloudFrontOriginAccessIdentity |
| GET Origin Access Identity List | ListCloudFrontOriginAccessIdentities |
| DELETE Origin Access Identity | DeleteCloudFrontOriginAccessIdentity |

# Policy Keys

Policy keys enable you to add conditions to your policies, such as request date or IP range. CloudFront implements the AWS-wide policy keys, but no others. For more information about policy keys, see "Condition" in the Element Descriptions section of *Using AWS Identity and Access Management*.

# Example Policies for CloudFront

This section shows a few simple policies for controlling User access to CloudFront.

> **Note**
>
> In the future, CloudFront might add new actions that should logically be included in one of the following policies, based on the policy's stated goals.

### Example 1: Allow a group read and write access to all of resources owned by the account

This example creates a policy that is attached to a group (for example, the Developers group) to give the group read and write access to all CloudFront resources.

```
{
   "Statement":[{
      "Effect":"Allow",
      "Action":["cloudfront:*"],
      "Resource":"*"
      }
   ]
}
```

### Example 2: Allow a group read access to all of resources owned by the account

This example creates a policy that is attached to a group (for example, the Finance group) to give the group read access to all CloudFront resources.

```
{
   "Statement":[{
      "Effect":"Allow",
      "Action":["cloudfront:Get*", "cloudfront:List*"],
      "Resource":"*"
      }
   ]
}
```

### Example 3: Allow a group read and write access to all distributions owned by the account

This example creates a policy that is attached to a group (for example, the Ops group) to give the group read and write access to all distributions, but not access to invalidations or origin access identities.

```
{
   "Statement":[{
      "Effect":"Allow",
      "Action":["cloudfront:*Distribution*"],
      "Resource":"*"
      }
   ]
}
```

**Example 4: Allow a group to retrieve CloudFront distribution data, but only if they're using SSL with the request**

This example creates a policy that is attached to a group to give the group access to all CloudFront actions, with a condition that requires use of SSL.

```
{
    "Statement":[{
        "Effect":"Allow",
        "Action":["cloudfront:*"],
        "Resource":"*",
        "Condition":{
            "Bool":{
                "aws:SecureTransport":"true"
            }
        }
    }
    ]
}
```

# Overall Flow for Using Amazon CloudFront

The following diagram and table explain the basic flow for using Amazon CloudFront with an Amazon S3 origin. It assumes you're already signed up for CloudFront and Amazon S3.
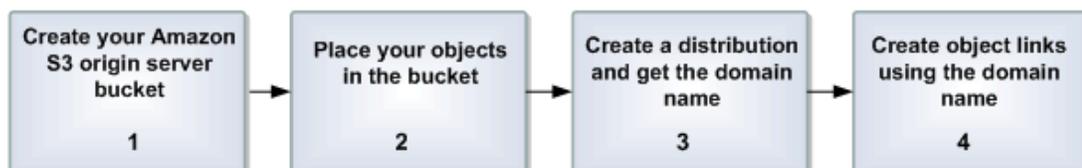


**Process for Using CloudFront**

| | |
|---|---|
| 1 | You create the Amazon S3 bucket that will be your origin server. |
| 2 | You place your objects in the bucket and make them publicly readable (if you want to serve content that isn't publicly readable, see Using a Signed URL to Serve Private Content (p. 70)). |
| 3 | You create your CloudFront distribution and get the domain name that CloudFront assigns.<br>Example distribution ID: `EDFDVBD6EXAMPLE`<br>Example domain name: `d111111abcdef8.cloudfront.net`<br>The distribution ID will not necessarily match the domain name. |
| 4 | You create the URLs that your end users will use to get your objects and include them as needed in your web application or website.<br>Example URL: `http://d111111abcdef8.cloudfront.net/images/image.jpg` |

If you want to use your own domain name instead of the CloudFront domain name in the URLs, you can create a CNAME alias for the CloudFront name. For more information, see Using CNAMEs (p. 49).

# Migrating from Amazon S3 to CloudFront

If you currently distribute content from your Amazon S3 bucket using a CNAME, you can migrate to CloudFront with no disruption. This section describes the process.
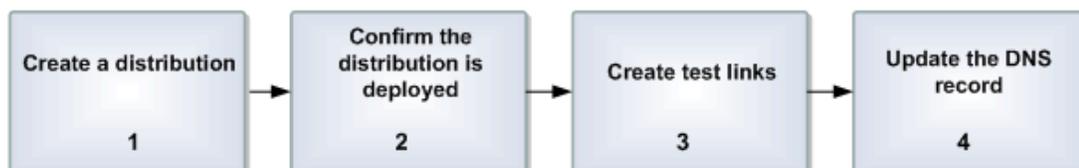


**Process for Migrating from Amazon S3 to CloudFront**

| 1 | Create a CloudFront distribution. You can use one of the various GUI-based CloudFront tools available, or use the CloudFront control API directly. For information about the tools, go to the Amazon CloudFront Developer Tools.<br><br>When you create the distribution, provide the name of your Amazon S3 bucket as the origin server (in the format *<bucket name>*.s3.amazonaws.com).<br><br>Also provide the CNAME you've been using with the bucket.<br><br>For more information, see Parts of a Distribution (p. 37) and go to POST Distribution in the *Amazon CloudFront API Reference*. |
|---|---|
| 2 | Confirm your distribution is fully deployed (i.e., that its status is `Deployed`).<br><br>GUI-based tools for CloudFront typically display the distribution's status.<br><br>For more information about getting a distribution's status with the CloudFront control API, go to GET Distribution in the *Amazon CloudFront API Reference*. |
| 3 | Create test links to publicly readable objects in your Amazon S3 bucket, and test the links.<br><br>Make sure to use the distribution's DNS name in the links. For example, `http://e604721fxaaqy9.cloudfront.net/images/image.jpg`.<br><br>For more information about the link format, see Format of URLs for CloudFront Objects (p. 21). |

| 4 | Update your existing DNS record to point to the distribution's domain name instead of the Amazon S3 bucket. |
|---|---|
| | The exact procedure for configuring DNS depends on your DNS server or DNS provider and is beyond the scope of this document. |

After you update the DNS record, the DNS system propagates your CNAME change throughout the DNS caches (a process that can take up to 72 hours, but usually happens faster). During this time, some requests for your content will resolve to the Amazon S3 bucket, and others to CloudFront.

To confirm that the switch from Amazon S3 to CloudFront has happened, you can use a DNS tool like *dig* to confirm that the CNAME points to the CloudFront distribution domain name, and not your Amazon S3 bucket (for information about dig, go to http://www.kloth.net/services/dig.php). You can also monitor the number of requests that are served from Amazon S3 versus CloudFront. The number of Amazon S3 requests should decrease, whereas it should increase for CloudFront. You can see the request numbers by viewing your AWS account activity or the AWS usage reports for Amazon S3 and CloudFront (for more information, see General Usage Data (p. 128)).

**Related Topics**

- Using CNAMEs (p. 49)

# Working with Objects

**Topics**

This section describes how you work with objects in the CloudFront system.

# Format of URLs for CloudFront Objects

**Topics**

## Public and Signed URLs

When you create a distribution, you receive the CloudFront domain name associated with that distribution. You use that domain name when creating the links to your objects. If you have another domain name that you'd rather use, you can use a CNAME alias. For more information, see Using CNAMEs (p. 49).

When you create URLs to give end users access to objects in your CloudFront distribution, the URLs are either public URLs or signed URLs:

*Public URLs* allow users to access the following objects:

- Objects in a public-content distribution.

- Objects in a private-content distribution that doesn't require signed URLs. In this configuration, end users must access the content through CloudFront; they can't directly access content in an Amazon S3 bucket. However, the CloudFront distribution has not been configured to require signed URLs.

*Signed URLs* are required to access objects in a private-content distribution that you have configured to require signed URLs.

You can configure a private-content distribution to use either public URLs or signed URLs, but not both. For more information, see Using a Signed URL to Serve Private Content (p. 70).

# Public URLs for Objects in Amazon S3

A public URL for an object in an Amazon S3 bucket uses this format:

```
http://<CloudFront domain name>/<object name in Amazon S3 bucket>.
```

> ⚠️ **Important**
>
> If the distribution serves streaming content, additional characters are required in the path to the file. For more information, see Configuring the Media Player (p. 60).

For example, suppose you have an Amazon S3 bucket called `mybucket`. The bucket contains a publicly readable object named `images/image.jpg`.

You create a CloudFront distribution and specify `mybucket.s3.amazonaws.com` as the origin server for this distribution. CloudFront returns `d111111abcdef8.cloudfront.net` as the domain name for the distribution and `EDFDVBD6EXAMPLE` as the distribution ID.

The URL you give to end users to access the object in this example is:

```
http://d111111abcdef8.cloudfront.net/images/image.jpg.
```

For more information about names and paths for Amazon S3 buckets, see Virtual Hosting of Buckets in the *Amazon Simple Storage Service Developer Guide*.

Anytime an end user accesses that object, CloudFront serves the object from the appropriate edge location. If the object isn't in that edge location, CloudFront goes to the origin server associated with the `EDFDVBD6EXAMPLE` distribution (`mybucket.s3.amazonaws.com`) and gets a copy of the object for the edge location to serve to the end user.

# Public URLs for Objects in a Custom Origin

The format of public URLs for objects in a custom origin are much like public URLs for objects in Amazon S3:

```
http://<CloudFront domain name>/<object name in custom origin>
```

If your object is in a folder on your origin server, then the CloudFront URL must include the name of the folder. For example, if `image.jpg` is located in the `images` folder, then the URL is:

```
http://d111111abcdef8.cloudfront.net/images/image.jpg
```

CloudFront gets objects from the domain that you specified when you created the distribution, using the object path and name in the public URL. For example, if the domain for your custom origin is `example.com` and the object path and name is `/images/image.jpg`, CloudFront will get the object from the following location:

```
http://example.com/images/image.jpg
```

## How Public URLs Affect the Invalidation of Directories

If you use CloudFront URLs that give end users access to directories, we recommend that you always use the same URL format, either with a trailing slash (/) or without, for example:

```
http://d111111abcdef8.cloudfront.net/images/
```

```
http://d111111abcdef8.cloudfront.net/images
```

Browsers and other web applications will resolve both formats to the same directory. However, CloudFront stores public URLs exactly as you specify them, and if you want to invalidate a directory, you'll need to specify the exact same directory, including or excluding the slash. If you don't have a standard for how you specify directories, you'll need to invalidate the directory with and without the slash to ensure that CloudFront removes the directory from the edge location. If you've reached the limit for free invalidations for the month, you'll pay for both invalidations even though only one of the directories exists.

## Signed URLs

Signed URLs allow end users to access objects in a distribution that is configured to serve private content. The URLs include extra information that restricts access to the cached objects. For information about the format of signed URLs, see Using a Signed URL to Serve Private Content (p. 70).

## How CloudFront Processes HTTP and HTTPS Requests

For Amazon S3 origins, CloudFront accepts requests in both HTTP and HTTPS protocols for objects in a CloudFront distribution by default. CloudFront then passes the requests to your Amazon S3 bucket or custom origin using the same protocol with which the requests were made.

For custom origins, when you create your distribution, you can specify how CloudFront accesses your origin: HTTP only, or matching the protocol that is used by the viewer. For more information about how CloudFront handles HTTP and HTTPS requests for custom origins, see Protocols (p. 66)

For information about how to restrict your download distribution so that end users can only access objects using HTTPS, see Creating Secure HTTPS Connections (p. 113). (This option doesn't apply to streaming distributions, which use the RTMP protocol.)

> **Note**
>
> The charge for HTTPS requests is higher than the charge for HTTP requests. For more information about billing rates, go to the CloudFront pricing plan.

## How CloudFront Processes and Logs Query String Parameters

If an application appends any query string parameters to the URL for a CloudFront object, CloudFront ignores the parameters when it identifies the object to serve. For example, all of the following URLs return the same object:

- `http://d111111abcdef8.cloudfront.net/images/image.jpg`
- `http://d111111abcdef8.cloudfront.net/images/image.jpg?parameter1=a`
- `http://d111111abcdef8.cloudfront.net/images/image.jpg?parameter1=a&parameter2=b`

When CloudFront requests an object from the origin server, it removes any query string parameters. If you enable logging, CloudFront logs every query string. For more information about CloudFront logging, see Download Distribution File Format (p. 123).

# Adding, Removing, or Replacing Objects in a Distribution

For information about adding objects to a distribution, see Adding Objects to a CloudFront Distribution (p. 24).

When you replace objects in your distribution, we recommend that you use versioned object names. For more information, see Updating Existing Objects Using Versioned Object Names (p. 24). You can also replace objects with objects that have the same name. See Updating Existing Objects Using the Same Object Names (p. 25). Regardless of how you choose to replace objects in your distribution, we recommend that you specify when objects should be removed from the CloudFront cache. For more information, see Specifying When an Object Expires (p. 25).

If you need to quickly remove objects from a distribution, you can invalidate them. For more information, see Invalidating Objects (p. 26).

## Adding Objects to a CloudFront Distribution

When you add an object to your origin, ensure that you're adding it to one of the Amazon S3 buckets in your distribution or, for a custom origin, to a directory in the specified domain.

When you add an object to your origin and expose a CloudFront link to the object, a CloudFront edge location won't fetch the object from the origin until the edge location receives an end-user request for the object.

CloudFront servers don't determine the MIME type for the objects they serve. When you upload an object to your origin, you should set the `Content-Type` header field for the object.

## Updating Existing Objects Using Versioned Object Names

When you update existing objects in a CloudFront distribution, we recommend that you include some sort of version identifier either in your object names or in your directory names to give yourself better control over your content. This identifier might be a date-time stamp, a sequential number, or some other method of distinguishing two versions of the same object.

For example, instead of naming a graphic file image.jpg, you might call it image_1.jpg. When you want to start serving a new version of the file, you'd name the new file image_2.jpg, and you'd update your links to point to image_2.jpg. Alternatively, you might put all graphics in an images_v1 directory and, when you want to start serving new versions of one or more graphics, you'd create a new images_v2 directory, and you'd update your links to point to that directory. With versioning, you don't have to wait for an object to expire before CloudFront begins to serve a new version of it, and you don't have to pay for object invalidation.

Even if you version your objects, we still recommend that you set an expiration date. For more information, see Specifying When an Object Expires (p. 25).

> **Note**
>
> Specifying versioned object names or directory names is not related to Amazon S3 object versioning.

# Updating Existing Objects Using the Same Object Names

Although you can update existing objects in a CloudFront distribution and use the same object names, we don't recommend it. CloudFront distributes objects to edge locations only when the objects are requested, not when you put new or updated objects in your origin. If you update an existing object in your origin with a newer version that has the same name, an edge location won't get that new version from your origin until both of the following occur:

- The old version of the object in the cache expires. For more information, see Specifying When an Object Expires (p. 25).
- There's an end user request for the object at that edge location.

If you use the same names when you replace objects, you can't control when CloudFront starts to serve the new files. By default, CloudFront caches objects in edge locations for 24 hours. (For more information, see Specifying When an Object Expires (p. 25).) For example, if you're replacing all of the objects on an entire website:

- Objects for the less popular pages may not be in any edge locations. The new versions of these objects will start being served on the next request.
- Objects for some pages may be in some edge locations and not in others, so your end users will see different versions depending on which edge location they're served from.
- New versions of the objects for the most popular pages might not be served for up to 24 hours because CloudFront might have retrieved the objects for those pages just before you replaced the objects with new versions.

# Specifying When an Object Expires

Typically, an object stays in an edge location until it expires. After it expires, the next time the edge location gets an end-user request for the object, CloudFront forwards the request to the origin server. (If an object in an edge location isn't frequently requested, CloudFront might evict the object—remove the object before its expiration date—to make room for objects that are more popular.)

By default, each object automatically expires after 24 hours. To specify a different expiration time, specify a value for either the `Cache-Control max-age` directive or the `Expires` header field on the object in the origin server:

- The `Cache-Control max-age` directive lets you specify how long (in seconds) you want the object to remain in the cache before CloudFront gets the object again from the origin server. The minimum expiration time CloudFront supports is 0 seconds. The maximum expiration time is unlimited. RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1, might recommend maximum values, but CloudFront allows any value.
- The `Expires` header field lets you specify an expiration date and time using the format specified in RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1 Section 3.3.1, Full Date, for example:

  ```
  Sat, 30 Jun 2012 23:59:59 GMT
  ```

Use the `Expires` header field carefully. After the expiration date and time passes, CloudFront gets the object again from the origin server every time an edge location receives a request for the object.

If you specify a value both for the `Cache-Control max-age` directive and for the `Expires` header field, the `max-age` directive takes precedence.

You cannot use the HTTP `Cache-Control` or `Pragma` header fields in a `GET` request from an end user to force CloudFront to go back to the origin server for the object. CloudFront ignores those header fields from the end user.

For more information about the `Cache-Control` and `Expires` header fields, see the RFC 2616 specification.

For an example of how to add `Cache-Control` and `Expires` header fields using the AWS SDK for PHP, see Upload an Object Using the AWS SDK for PHP in the *Amazon Simple Storage Service Developer Guide.* Some third-party tools are also able to add these fields.

To add a `Cache-Control` or `Expires` header field for an Amazon S3 object using the Amazon S3 console

1.  Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2.  In the Amazon S3 console, in the Buckets pane, click the name of the bucket that contains the files.
3.  At the top of the right pane, click **Actions**. In the Actions list, click **Properties**.
4.  In the Properties pane, click the **Metadata** tab.
5.  In the Objects and Folders pane, click the name of a file for which you want to add a `Cache-Control` or `Expires` header field.
6.  On the Metadata tab, click **Add More Metadata**.
7.  In the Key list, click **Cache-Control** or `Expires`, as applicable.
8.  In the **Value** field, enter the applicable value:

    *   For a `Cache-Control` field, enter a number of seconds.
    *   For an `Expires` field, enter a date and time in HTML format.

9.  Click **Save**.
10. Repeat Steps 5 through 9 for the remaining files.

# Invalidating Objects

If you need to remove an object from CloudFront edge-server caches before it would expire, you can do one of the following:

*   Invalidate the object. The next time an end user requests the object, CloudFront returns to the origin to fetch the latest version of the object.
*   Use object versioning to serve a different version of the object that has a different name. For more information, see Updating Existing Objects Using Versioned Object Names (p. 24).

You can invalidate a specified number of objects each month for free. Above that limit, you pay a fee for each invalidation request; each request invalidates one object. For example, to invalidate a directory and all of the files in the directory, you must invalidate the directory and each file individually. If you need to

invalidate a lot of files, it may be easier and less expensive to create a new distribution and change your object paths to refer to the new distribution. For more information about the charges for invalidation, see Paying for Object Invalidation (p. 28).

The CloudFront console doesn't currently allow you to invalidate objects. To invalidate objects using CloudFront, you must either submit an invalidation batch request to CloudFront or use a third-party tool that provides a graphical user interface for invalidating objects. For information about using the CloudFront API, see Invalidating Objects Using the CloudFront API (p. 27). For information about third-party tools, see Third-Party Tools for Invalidating Objects (p. 27)

# Choosing Between Invalidating Objects and Using Versioned Object Names

To control the versions of objects served from your distribution, you can either invalidate objects or give them versioned file names. If you'll want to update your objects frequently, we recommend that you primarily use object versioning for the following reasons:

- Versioning enables you to control which object a request returns even when the end user has a version cached either locally or behind a corporate caching proxy. If you invalidate the object, the end user may continue to see the old version until it expires from those caches.
- File names are included in the CloudFront access logs, so versioning makes it easier to analyze the results of object changes.
- Versioning provides a way to serve different versions of objects to different end users.
- Versioning simplifies rolling forward and backward between object revisions.
- Versioning is less expensive. You still have to pay for CloudFront to transfer new versions of your objects to edge locations, but you don't have to pay the per-file charge for invalidating objects.

For more information about object versioning, see Updating Existing Objects Using Versioned Object Names (p. 24).

# Invalidating Objects Using the CloudFront API

For information about creating invalidation requests using the CloudFront API, go to POST Invalidation.

In addition, CloudFront includes the following API actions for getting information about invalidation requests:

- Getting a list of your invalidation requests: GET Invalidation List)
- Getting a specific invalidation request: GET Invalidation)

# Third-Party Tools for Invalidating Objects

In addition to the invalidation methods provided by CloudFront, you can use the following third-party tools to invalidate objects.

> **Note**
>
> These tools were developed by third-party vendors who are not associated with Amazon Web Services. For information on how to use these tools, please refer to the vendor's documentation or contact the vendor.

- CloudBuddy Personal, http://m1.mycloudbuddy.com/index.html
- CloudBerry Explorer, http://cloudberrylab.com

- Ylastic, http://ylastic.com

- Cyberduck, http://cyberduck.ch

- Bucket Explorer, http://www.bucketexplorer.com

- CloudFront Invalidator, http://www.swook.net/p/cloudfront-invalidator.html

- CDN Planet CloudFront Purge Tool,
  http://www.cdnplanet.com/tools/cloudfront-purge-tool/http://www.cdnplanet.com/tools/cloudfront-purge-tool/

You can also search for code samples on Github, https://github.com. Search on the phrase "CloudFront invalidation".

## Invalidation Limits

You can make any number of invalidation requests, but you can have only three invalidation requests per distribution in progress at one time. Each request can contain up to 1,000 objects to invalidate. If you exceed these limits, CloudFront returns an error message.

> 📕 **Note**
>
> It usually takes 10 to 15 minutes for CloudFront to complete your invalidation request, depending on the size of the request.

## Paying for Object Invalidation

The first 1,000 object invalidations you request per month are free; you pay for each object invalidation over 1,000 in a month. This limit applies to the total number of object invalidations across all of the distributions that you create with one AWS account. For example, if you use the AWS account `john@example.com` to create three distributions, and each distribution has 600 object invalidations in a given month (for a total of 1,800 invalidations), AWS will charge you for 800 object invalidations in that month. For specific information about invalidation pricing, go to Amazon CloudFront Pricing.

> 📕 **Note**
>
> For the purposes of invalidation pricing, an object invalidation request is defined as a single `Path` element object. For more information about the `Path` element, see Invalidating Objects Using the CloudFront API (p. 27).

# How CloudFront Processes Partial Requests for an Object (Range GETs)

For large objects, an end user's browser or client might make multiple `GET` requests and use the `Range` request header to download the object in smaller units. These requests for ranges of bytes, sometimes known as `Range GET` requests, improve the efficiency of partial downloads and the recovery from partially failed transfers. CloudFront follows the RFC specification for the `Range` header. For more information, see "Section 14.35 Range" in *Hypertext Transfer Protocol -- HTTP/1.1* at http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35.

# Setting the Content-Type Header When You Upload Objects to Your Origin

CloudFront servers don't determine the MIME type for the objects they serve. Therefore, when you upload an object to your origin, you should set the `Content-Type` header for the object.

# Specifying a Default Root Object

You can configure CloudFront to return a specific object (the default root object) when an end user requests the root URL for your distribution instead of an object in your distribution. Specifying a default root object avoids exposing the contents of your distribution.

For example, the following request points to the object `image.jpg`:
`http://d111111abcdef8.cloudfront.net/image.jpg`

The following request points to the root URL of the same distribution instead of to a specific object:
`http://d111111abcdef8.cloudfront.net/`

When you define a default root object, an end-user request that calls the root of your distribution returns the default root object. For example, if you designate the file `index.html` as your default root object, a request for:
`http://d111111abcdef8.cloudfront.net/`
returns:
`http://d111111abcdef8.cloudfront.net/index.html`

However, if you define a default root object, an end-user request for a subdirectory of your distribution does not return the default root object. For example, suppose `index.html` is your default root object and that CloudFront receives an end-user request for the `install` directory under your CloudFront distribution:
`http://d111111abcdef8.cloudfront.net/install/`

CloudFront will not return the default root object even if a copy of `index.html` appears in the `install` directory.

The behavior of CloudFront default root objects is different from the behavior of Amazon S3 index documents. When you configure an Amazon S3 bucket as a website and specify the index document, Amazon S3 returns the index document even if a user requests a subdirectory in the bucket. (A copy of the index document must appear in every subdirectory.) For more information about configuring Amazon S3 buckets as websites and about index documents, see the Hosting Websites on Amazon S3 chapter in the *Amazon Simple Storage Service Developer Guide*.

⚠️ **Important**

Remember that a default root object applies only to your CloudFront distribution. You still need to manage security for your origin. For example, if you are using an Amazon S3 origin, you still need to set your Amazon S3 bucket ACLs appropriately to ensure the level of access you want on your bucket.

If you don't define a default root object, requests for the root of your distribution pass to your origin server. If you are using an Amazon S3 origin, any of the following might be returned:

- **A list of the contents of your Amazon S3 bucket**—Under any of the following conditions, the contents of your origin are visible to anyone who uses CloudFront to access your distribution:

- Your bucket is not properly configured.

- The Amazon S3 permissions on the bucket associated with your distribution and on the objects in the bucket grant access to *everyone*.

- An end user accesses your origin using your origin root URL.

- **A list of the private contents of your origin**—If you configure your origin as a private distribution (only you and CloudFront have access), the contents of the Amazon S3 bucket associated with your distribution are visible to anyone who has the credentials to access your distribution through CloudFront. In this case, users are not able to access your content through your origin root URL. For more information about distributing private content, see Using a Signed URL to Serve Private Content (p. 70).

- **Error 403 Forbidden**—CloudFront returns this error if the permissions on the Amazon S3 bucket associated with your distribution or the permissions on the objects in that bucket deny access to CloudFront and to everyone.

To avoid exposing the contents of your distribution or returning an error, you can specify a default root object for your distribution.

### To specify a default root object for your distribution

1. Upload the default root object to the origin that your distribution points to.

   The file can be any type supported by CloudFront. For a list of constraints on the file name, see the description of the `DefaultRootObject` element in DistributionConfig Complex Type.

   > **Note**
   >
   > If the file name of the default root object is too long or contains an invalid character, CloudFront returns the error `HTTP 400 Bad Request - InvalidDefaultRootObject`. In addition, CloudFront caches the code for five minutes and writes the results to the access logs.

2. Confirm that the permissions for the object grant CloudFront at least `read` access.

   For more information about Amazon S3 permissions, see Access Control in the *Amazon Simple Storage Service Developer Guide*. For information on using the Amazon S3 console to update permissions, see the *Amazon Simple Storage Service Console User Guide*.

3. Update your distribution to refer to the default root object. You can use the AWS Management Console to update your distribution, or you can use the CloudFront API.

   To update your configuration using the AWS Management Console:

   a. Sign in to the AWS Management Console and open the Amazon CloudFront console at https://console.aws.amazon.com/cloudfront/.

   b. Select the distribution to update.

   c. Click **Edit**.
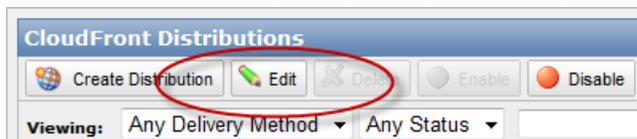
   

   d. In the Edit Distribution dialog box, enter the file name of the default root object in the **Default Root Object** field.

e.   To save your changes, click **Yes, Edit**.

To update your configuration using the CloudFront API, you add the `DefaultRootObject` element to the distribution. The following example shows a CloudFront distribution configuration with the `DefaultRootObject` element, and with `index.html` designated as the default root object.

```
<DistributionConfig
 xmlns="http://cloudfront.amazonaws.com/doc/2012-03-15/">
    <S3Origin>
        <DNSName>myawsbucket</DNSName>
    </S3Origin>
    <CallerReference>20120229090000</CallerReference>
 <Comment>My comments</Comment>
 <Enabled>true</Enabled>
 <DefaultRootObject>index.html</DefaultRootObject>
 <Logging>
  <Bucket>mylogs.s3.amazonaws.com</Bucket>
  <Prefix>myprefix/</Prefix>
 </Logging>
</DistributionConfig>
```

For more information about the configuration for a distribution, see Updating a Distribution's Configuration (p. 53). For more information about the `DefaultRootObject` element, go to DistributionConfig Complex Type in the *Amazon CloudFront API Reference*.

4.   Confirm that you have enabled the default root object by requesting your root URL. If your browser doesn't display the default root object, perform the following steps:

a.   Confirm that your distribution is fully deployed by viewing the status of your distribution in the Amazon CloudFront console.

b.   Repeat Steps 2 and 3 to verify that you granted the correct permissions and that you correctly updated the configuration of your distributiuon to specify the default root object.

# Serving Compressed Files

Amazon CloudFront can serve both compressed and uncompressed files from an origin server. CloudFront relies on the origin server either to compress the files or to have compressed and uncompressed versions of files available; CloudFront does not perform the compression on behalf of the origin server. With some qualifications, CloudFront can also serve compressed content from Amazon S3. For more information, see Choosing the File Types to Compress (p. 33).

Serving compressed content makes downloads faster because the files are smaller—in some cases, less than half the size of the original. Especially for JavaScript and CSS files, faster downloads translates into faster rendering of web pages for your users. In addition, because the cost of CloudFront data transfer is based on the total amount of data served, serving compressed files is less expensive than serving uncompressed files.

CloudFront can only serve compressed data if the viewer (for example, a web browser or media player) requests compressed content by including `Accept-Encoding: gzip,deflate` in the request header. The content must be compressed using gzip or deflate; other compression algorithms are not supported. If the request header includes additional content encodings, for example, `sdch`, CloudFront removes them before forwarding the request to the origin server. If either `gzip` or `deflate` is missing from the `Accept-Encoding` field, CloudFront serves only the uncompressed version of the file. For more information about the `Accept-Encoding` request-header field, see "Section 14.3 Accept Encoding" in *Hypertext Transfer Protocol -- HTTP/1.1* at http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html.

## How CloudFront Serves Compressed Content from a Custom Origin

Here's how CloudFront commonly serves compressed content from a custom origin to a web application:

1. You configure your web server to compress selected file types. For more information, see Choosing the File Types to Compress (p. 33).

2. You create a CloudFront distribution.

3. You program your web application to access files using CloudFront URLs.

4. A user accesses your application in a web browser.

5. CloudFront directs web requests to the edge location that has the lowest latency for the user, which may or may not be the geographically closest edge location.

6. At the edge location, CloudFront checks the cache for the object referenced in each request. If the browser included `Accept-Encoding: gzip,deflate` in the request header, CloudFront checks for a compressed version of the file. If not, CloudFront checks for an uncompressed version.

7. If the file is in the cache, CloudFront returns the file to the web browser. If the file is not in the cache:

   a. CloudFront forwards the request to the origin server.

   b. If the request is for a type of file that you want to serve compressed (see Step 1), the web server compresses the file.

   c. The web server returns the file (compressed or uncompressed, as applicable) to CloudFront.

   d. CloudFront adds the file to the cache and serves the file to the user's browser.

# Serving Compressed Files When Your Origin Server Is Running IIS

By default, IIS does not serve compressed content for requests that come through proxy servers such as CloudFront. If you're using IIS and if you configured IIS to compress content by using the `httpCompression` element, change the values of the `noCompressionForHttp10` and `noCompressionForProxies` attributes to false.

# Serving Compressed Files from Amazon S3

If you want to serve compressed files from Amazon S3:

1. Create two versions of each file, one compressed and one uncompressed. To ensure that the compressed and uncompressed versions of a file don't overwrite one another in the CloudFront cache, give each file a unique name, for example, welcome.js and welcome.js.gz.

2. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.

3. Upload both versions to Amazon S3.

4. Add a `Content-Encoding` header field for each compressed file and set the field value to `gzip` or `deflate`, as applicable.

   For an example of how to add a `Content-Encoding` header field using the AWS SDK for PHP, see Upload an Object Using the AWS SDK for PHP in the *Amazon Simple Storage Service Developer Guide*. Some third-party tools are also able to add this field.

   To add a `Content-Encoding` header field and set the field value using the Amazon S3 console, perform the following procedure:

   a. In the Amazon S3 console, in the Buckets pane, click the name of the bucket that contains the compressed files.

   b. At the top of the right pane, click **Actions** and, in the **Actions** list, click **Properties**.

   c. In the Properties pane, click the **Metadata** tab.

   d. In the Objects and Folders pane, click the name of a file for which you want to add a `Content-Encoding` header field.

   e. On the Metadata tab, click `Add More Metadata`.

   f. In the Key list, click `Content-Encoding`.

   g. In the Value field, enter `gzip` or `deflate`, as applicable.

   h. Click **Save**.

   i. Repeat Step 3e through 3i for the remaining compressed files.

5. When generating HTML that links to content in CloudFront (for example, using php, asp, or jsp), evaluate whether the request from the viewer includes `Accept-Encoding: gzip,deflate` in the request header. If so, rewrite the corresponding link to point to the compressed object name.

# Choosing the File Types to Compress

Some types of files compress well, for example, HTML, CSS, and JavaScript files. Some types of files may compress a few percent, but not enough to justify the additional processor cycles required for your web server to compress the content, and some types of files even get larger when they're compressed. File types that generally don't compress well include graphic files that are already compressed (.jpg, .gif),

video formats, and audio formats. We recommend that you test compression for the file types in your distribution to ensure that there is sufficient benefit to compression.

# Restricting Access to Objects Based on the Geographic Location of End Users (Geoblocking)

You can restrict access to your objects based on the geographic location of your end users by using CloudFront's private-content feature along with a third-party geolocation service. To restrict access, you make your CloudFront distribution private. When the end user requests an object in your private distribution, you send the user's IP address to the geolocation service of your choice. Based on the geographic location returned by the third-party geolocation service for that end user, your web application either generates a CloudFront signed URL for the user or displays a message explaining why the user isn't allowed to access the object.

For more information about geoblocking, see the tutorial Restricting Access to Files in a CloudFront Distribution Based on Geographic Location (Geoblocking) (p. 178). For more information about CloudFront private content, see Using a Signed URL to Serve Private Content (p. 70).

# Working with Distributions

**Topics**

This section describes how you work with, configure, and manage a *distribution*. For a basic description of a distribution, see Distributions (p. 4). For a basic description of streaming distributions, see Streaming Media on Demand (p. 56).

# Types of Distributions

CloudFront offers different types of distributions to serve different types of content. Your AWS account can have up to 100 distributions of each type. There's no limit on the number of files you can serve per distribution.

The following table describes the distribution types and summarizes the different CloudFront features available with each type.

| Type | Description | Features Available |
|------|-------------|--------------------|
| Download | Serves non streaming files over an HTTP(S) connection. For example: image files, CSS files, JavaScript files, etc. <br><br> For information about creating a download distribution using an Amazon S3 origin, go to the Amazon CloudFront Getting Started Guide. For information about creating a download distribution with a custom origin, see Creating a Distribution with a Custom Origin (p. 41). | • Access logs <br> • CNAMEs <br> • Custom origins <br> • Private content <br> • Public content |
| Streaming | Streams media files over a Real-Time Messaging Protocol (RTMP) connection. For information, see Streaming Media on Demand (p. 56). | • Access logs <br> • CNAMEs <br> • Private content <br> • Public content |

*CNAMEs* means that you can use your own domain name instead of the CloudFront distribution name to serve the content (for more information, see Using CNAMEs (p. 49)).

*Public content* means that the content in your origin must be publicly readable.

*Private content* means that you can mark the content in the origin as readable only by you and CloudFront (for more information, see Using a Signed URL to Serve Private Content (p. 70)).

*Access logs* means that you can get a log with information about end-user access to your objects in the edge locations (for more information, see Access Logs (p. 117)).

There's a separate set of CloudFront control API actions for each type of distribution. The actions for download distributions act on the `distribution` resource; whereas the actions for streaming distribution act on the `streaming-distribution` resource. For a list of the actions, see Actions on Distributions (p. 36).

# Actions on Distributions

Each distribution has basic metadata and a configuration object associated with it. The following table lists the actions you can perform on a distribution or its configuration object and provides links to the corresponding control API reference topics. The table has one column for download distributions (which use an HTTP connection), and another column for streaming distributions (which use an RTMP connection).

| Action | API Reference Topic for Download Distributions | API Reference Topic for Streaming Distributions |
|--------|------------------------------------------------|-------------------------------------------------|
| Create a distribution | Go to POST Distribution | Go to POST Streaming Distribution |
| List your distributions | Go to GET Distribution List | Go to GET Streaming Distribution List |

| Action | API Reference Topic for Download Distributions | API Reference Topic for Streaming Distributions |
|---|---|---|
| Get a distribution's information (both the metadata and the configuration object) | Go to GET Distribution | Go to GET Streaming Distribution |
| Get only a distribution's configuration object | Go to GET Distribution Config | Go to GET Streaming Distribution Config |
| Update a distribution's configuration object | Go to PUT Distribution Config | Go to PUT Streaming Distribution Config |
| Delete a distribution | Go to DELETE Distribution | Go to DELETE Streaming Distribution |

# Parts of a Distribution

**Topics**

This section describes the parts of a distribution.

## Parts You Provide

To have CloudFront recognize your objects, you create a distribution by using the CloudFront *control API* (for more information about creating a download distribution, go to POST Distribution in the *Amazon CloudFront API Reference*).

When you create any type of distribution, you must specify three things:

- The origin server
- Caller reference
- Whether the distribution should be enabled or disabled upon creation

You can optionally provide these other components:

- One or more CNAME aliases for the distribution's domain name
- Comments about the distribution
- Whether access logs are written for the distribution (download distributions only)
- A *CloudFront origin access identity* to use with the distribution (required for CloudFront to fetch private objects from an Amazon S3 bucket)
- Any trusted signers (required to create a *signed URL*, which controls whether the end user has access to the cached object)

When you create the distribution with the CloudFront control API, you provide an XML document that contains this information (for download distributions, it's in an element called `DistributionConfig`; for streaming distributions, it's `StreamingDistributionConfig`). For more information about download

distributions, go to DistributionConfig Complex Type in the *Amazon CloudFront API Reference*. For more information about streaming distributions, go to Streaming DistributionConfig Complex Type in the *Amazon CloudFront API Reference*.

# The Origin Server

When you create the distribution, you specify the origin server either as an Amazon S3 bucket, or as a custom origin.

## Amazon S3 Bucket

When you use Amazon S3 as your origin, you place any objects you want delivered through CloudFront in an Amazon S3 bucket. How you get your objects into Amazon S3 doesn't matter (you might use the Amazon S3 API or another tool). You can create a hierarchy in your bucket to store the objects, just like you would with any other Amazon S3 bucket. You incur regular Amazon S3 charges for storing the objects in the bucket (for more information about the charges to use CloudFront, see Paying for CloudFront (p. 11)).

> **Note**
>
> Using an existing Amazon S3 bucket as your CloudFront origin server doesn't change that bucket in any way; you can still use it as you normally would to store and access Amazon S3 objects (at the normal Amazon S3 prices).

You specify the Amazon S3 bucket's name using this format: *<bucket name>*.s3.amazonaws.com. Do *not* use the Amazon S3 path style for specifying the bucket, which is s3.amazonaws.com/*<bucket name>*. If the bucket has a CNAME alias, you must specify the bucket name and *not* the CNAME alias.

For more information about creating a distribution with an Amazon S3 origin, see Start Using CloudFront with Amazon S3 in the *Amazon CloudFront Getting Started Guide*.

> **Important**
>
> In the early days of Amazon S3, you could create DNS-incompatible bucket names. For your bucket to work with CloudFront, it must conform to DNS requirements. For more information, go to Bucket Restrictions and Limitations in the *Amazon Simple Storage Service Developer Guide*.

## Custom Origin

A custom origin is any origin server that isn't hosted on Amazon S3. A custom origin might be located on a set of servers you manage privately, or it could be an Amazon EC2 instance.

When you use a custom origin, you specify the origin server as a DNS name, along with the port and protocol policy for the origin. To learn how to create a distribution with a custom origin, see Creating a Distribution with a Custom Origin (p. 41).

You can use most CloudFront features with a custom origin, but there are a few exceptions. Currently, the following features are not available with custom origins:

* **Streaming distributions**—Not supported.

* **Private content**—Although you can use a signed URL to distribute content from a custom origin, for CloudFront to access the custom origin, the origin must remain publicly accessible.

For information about requirements and recommendations when using custom origins, see Requirements and Recommendations for Using Amazon EC2 and Other Custom Origins (p. 48).

### Distributions per Origin Server

You typically create one distribution per origin server, although you could create multiple distributions per origin server. For example, if you had two distributions for one origin server, you could reference a single object using either distribution. In this case, if you had an image file called `image.jpg` in your origin server, CloudFront would work with each distribution as though it referenced an individual `image.jpg` object: one `image.jpg` accessible through one distribution, and another `image.jpg` accessible through the other distribution.

> **Note**
>
> After you create a distribution, you can't change its origin server. If you need to change the origin server for a distribution, you must create a new distribution that uses the new origin server and update either your links or your DNS records to use the new distribution's domain name. You can then delete the original distribution (for more information, see Deleting a Distribution (p. 54)).

## Caller Reference

The *caller reference* is a unique value that you provide and CloudFront uses to prevent replays of your request. You must provide a new caller reference value and other new information in the request for CloudFront to create a new distribution. You could use a time stamp for the caller reference (for example: 20120229090000).

If you pass the same caller reference value and the rest of the request is the same, CloudFront doesn't create a new distribution. Instead, it returns information about the distribution you previously created with that caller reference.

If you pass the same caller reference value, but vary other information in the request, CloudFront returns a `DistributionAlreadyExists` error (for more information about errors, see Error Responses (p. 133)).

After you create a distribution, you can't change its caller reference.

## Enabled or Disabled

When you create a distribution, you must tell CloudFront if you want the distribution to be enabled or disabled once it's created. *Enabled* means that as soon as the distribution is fully deployed you can deploy links that use the distribution's domain name and end users can retrieve content. In other words, whenever a distribution is enabled, CloudFront accepts and handles any end-user requests for content that use the domain name associated with that distribution. For more information about full deployment, see Eventual Consistency (p. 4).

*Disabled* means that even though the distribution might be deployed and ready to use, end users can't use it. In other words, whenever a distribution is disabled, CloudFront doesn't accept any end-user requests that use the domain name associated with that distribution. Until you switch the distribution from disabled to enabled (by updating the distribution's configuration), no one can use it.

You can toggle a distribution between disabled and enabled as often as you like. Make sure to follow the process for updating a distribution's configuration (for more information, see Updating a Distribution's Configuration (p. 53)).

## CNAME Aliases

You can optionally associate one or more CNAME aliases with a distribution so that you can use a domain name of your choice in links to your objects instead of the domain name CloudFront assigns. For more information, see Using CNAMEs (p. 49).

## Comments

When you create a distribution, you can provide optional comments about the distribution. You can update those comments at any time. The maximum allowed length is 128 characters.

## Access Logs

This component applies only to download distributions.

When you create a download distribution, you can enable logging for the distribution. This means that CloudFront records information about each end user request for an object and stores the files in an Amazon S3 bucket of your choice. You can turn logging on or off for the download and streaming distributions at any time. For more information, see Access Logs (p. 117).

## Origin Access Identity and Trusted Signers

When you create a distribution, you can:

- Use the *origin access identity* to configure the distribution so that end users can *only* access objects in an Amazon S3 bucket through CloudFront
- Use *trusted signers* to configure the distribution so that you control end-user access to cached objects (through the use of a signed URL)

For more information about origin access identity and trusted signers, see Using a Signed URL to Serve Private Content (p. 70).

# Parts Amazon CloudFront Provides

When you create a new distribution, CloudFront returns the following information:

- The distribution's ID (for example, `EDFDVBD6EXAMPLE`)
- The distribution's domain name (for example, `d111111abcdef8.cloudfront.net`)
- The distribution's current status
- When the distribution was last modified
- A list of the *active trusted signers* for the distribution

## Distribution ID

The distribution's ID *won't necessarily* match the domain name. You can refer to the distribution by its ID (e.g., `EDFDVBD6EXAMPLE`). Whenever you perform an action on the distribution, you provide its fully qualified ID. For example, `https://cloudfront.amazonaws.com/2012-03-15/distribution/EDFDVBD6EXAMPLE`. For streaming distributions, the fully qualified ID looks like: `https://cloudfront.amazonaws.com/2012-03-15/streaming-distribution/EGTXBD79EXAMPLE`.

## Domain Name

You use the distribution's domain name in the links to your objects, unless you're using CNAME records. (For more information, see Using CNAMEs (p. 49)). For example, if your distribution's domain name is `d111111abcdef8.cloudfront.net`, the link to the example `images/image.jpg` file would be `http://d111111abcdef8.cloudfront.net/images/image.jpg`.

**Note**

CloudFront domain names are unique. Your distribution's domain name was never used for a previous distribution and will never be reused for another distribution in the future.

## Status

The possible status values for a distribution are listed in the following table.

| Value | Description |
| --- | --- |
| InProgress | The distribution is still being created or updated. |
| Deployed | The distribution has been created or updated and the changes have been fully propagated through the CloudFront system. |

**Note**

Even if the distribution's status is `Deployed`, you still must enable the distribution for use before end users can retrieve content. For more information, see Enabled or Disabled (p. 39).

## Last Modification Date

The last modification time stamp uses the ISO 8601 format (e.g., 2009-11-19T19:37:58Z). For more information, go to http://www.w3.org/TR/NOTE-datetime.

## Active Trusted Signers

CloudFront includes a list of the *active trusted signers* for the distribution. This information is included as part of the distribution only if you've set up the distribution to serve private content with a signed URL (for more information, see Using a Signed URL to Serve Private Content (p. 70)). Active trusted signers are *trusted signers* who have at least one active key pair that CloudFront is aware of. Only active trusted signers can create working signed URLs.

# Creating a Distribution with a Custom Origin

You can create a distribution that uses either a custom origin or an Amazon S3 origin. This section describes how to create a distribution with a custom origin. To learn how to create a distribution with an Amazon S3 origin, see Start Using CloudFront with Amazon S3 in the *Amazon CloudFront Getting Started Guide*.

For a basic description of origins and the features supported for use with custom origins, see The Origin Server (p. 38).

The following table describes the overall process for creating a distribution with a custom origin.

**Process for Creating a Distribution with a Custom Origin**

| | |
| --- | --- |
| 1 | Review the section called "Requirements and Recommendations for Using Amazon EC2 and Other Custom Origins" (p. 48). |
| 2 | Upload your content to your origin server and verify that it is accessible. |

| 3 | Create an Amazon CloudFront distribution. |
|---|---|
| 4 | Use the Amazon CloudFront domain name to reference content in your web pages or applications. |

# Upload Your Content to Your Origin Server

The content you deliver with CloudFront will be stored on a server referred to as an *origin server*. The origin server might be an Amazon Elastic Compute Cloud (EC2) instance, or it might be a server you manage privately.

Content can consist of just about any type of object, including images and videos. Before creating your Amazon CloudFront distribution, load the content you want to distribute on your custom origin server. You should also make sure that the content is accessible.

> **Caution**
>
> You are responsible for ensuring the security of your origin server. You must ensure that CloudFront has permission to access the server, and you must be sure that the security settings are appropriate to safeguard your content.

# Create an Amazon CloudFront Distribution

**Topics**
- Using the AWS Management Console to Create a Distribution (p. 42)
- Using the CloudFront API to Create a Distribution (p. 45)

To create a distribution with a custom origin, you can use either the AWS Management Console or the CloudFront API. The following sections describe both methods.

## Using the AWS Management Console to Create a Distribution

**To create an Amazon CloudFront distribution**

1.  Sign in to the AWS Management Console and open the Amazon CloudFront console at https://console.aws.amazon.com/cloudfront/.
2.  Click **Create Distribution**.
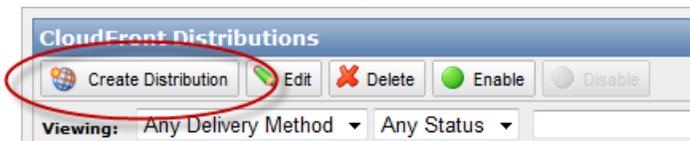


3.  From the **Create Distribution Wizard**, choose **Download** and **Custom Origin**.

> **Note**
>
> Currently, you cannot use a custom origin with a streaming distribution. CloudFront supports only Amazon S3 origin servers for streaming distributions.

4. Under **Custom Origin**, specify the following information:

   a. For **Origin DNS Name**, enter the origin to associate with the distribution. For example, www.example.com.

   b. Select the **Protocol Policy** to apply to your origin. If you select **HTTP Only**, CloudFront uses only HTTP to access the origin. If you specify **Match Viewer**, CloudFront fetches data from your origin using HTTP or HTTPS, based on the protocol of the viewer request.

   c. Specify which **HTTP Port** the custom origin listens on. The default value is port 80.

   d. If you selected Match Viewer as the Protocol Policy, you must also specify the **HTTPS Port** the custom origin listens on. The default value is port 443.

5. Click **Continue**.

6.  Next, specify the distribution details:

    a.  For **Allowed Connections**, to restrict access to your distribution to only HTTPS requests, select
        **HTTPS Only**. Otherwise, select **HTTP and HTTPS** to use either protocol to serve the request.
        For more information, see Creating Secure HTTPS Connections (p. 113).

    b.  If your distribution will use a CNAME, for **CNAMEs**, enter the CNAME alias you want to associate
        with this distribution. You can have up to 10 CNAME aliases per distribution. For more information,
        see Using CNAMEs (p. 49).

    c.  If you want to use a default root object with your distribution, for **Default Root Object** enter the
        default root object to associate with the distribution. For example, `index.html`. For more
        information, see Specifying a Default Root Object (p. 29).

    d.  If you want to enable logging, for **Logging** select **On**, and then from **Select Log Bucket**, select
        the Amazon S3 bucket to which you want to save your logs. Alternatively, you can type the
        bucket name in the **Specify Log Bucket** field. Type the log file prefix in the **Log Prefix** field.
        For more information, see Access Logs (p. 117).

    e.  In **Comments**, you can enter any comments you want to include about the distribution.

    f.  For the **Distribution Status**, select **Enabled** if you want the distribution to accept end-user
        requests for content as soon as it is deployed. Otherwise, if you prefer to enable the distribution
        later, choose **Disabled**. For more information, see Enabled or Disabled (p. 39).

7.  To review your distribution settings, click **Continue**.

8.  If you are satisfied with the distribution settings, click **Create Distribution**.

After creating the distribution, it might take a few minutes for the distribution to deploy. The distribution's
current status is displayed in the console under **Status**. **InProgress** indicates that the distribution is not
yet fully deployed.

> 📝 **Note**
>
> Even if the distribution's status is **Deployed**, you still must enable the distribution for use before end users can retrieve content. For more information, see Enabled or Disabled (p. 39).

When your distribution is deployed, you are ready to reference your content with your new Amazon CloudFront URL or CNAME, as described in Reference Content in Your Web Pages or Applications (p. 47).

# Using the CloudFront API to Create a Distribution

**To create an Amazon CloudFront distribution**

- Send a `POST` request on the `2012-03-15/distribution` resource. The request body must include an XML document with a `DistributionConfig` element.

The response echoes the `DistributionConfig` element and returns other metadata about the distribution. (For more information about the parts of a distribution, see Parts of a Distribution (p. 37).)

For example, you would post the following request and receive the following response. The request and response elements are described in DistributionConfig Complex Type in the *Amazon CloudFront API Reference*.

**Example Request**

```
POST /2012-03-15/distribution HTTP/1.1
Host: cloudfront.amazonaws.com
Authorization: [AWS authentication string]
Date: Thu, 19 Nov 2009 19:37:58 GMT
[Other required headers]

<?xml version="1.0" encoding="UTF-8"?>
<DistributionConfig xmlns="http://cloudfront.amazonaws.com/doc/2012-03-15/">
   <CustomOrigin>
       <DNSName>www.example.com</DNSName>
       <HTTPPort>80</HTTPPort>
       <HTTPSPort>443</HTTPSPort>
       <OriginProtocolPolicy>match-viewer</OriginProtocolPolicy>
   </CustomOrigin>
   <CallerReference>20091130090000</CallerReference>
   <CNAME>beagles.com</CNAME>
   <Comment>My comments</Comment>
   <Enabled>true</Enabled>
   <Logging>
       <Bucket>myawslogbucket.s3.amazonaws.com</Bucket>
       <Prefix>myprefix/</Prefix>
   </Logging>
</DistributionConfig>
```

For information about the headers required with every request, see REST Requests (p. 130).

**Example Response**

```
HTTP/1.1 201 Created
201 Created
```

```
Location: [URI of new distribution]
x-amz-request-id: [Request ID]

<?xml version="1.0" encoding="UTF-8"?>
<Distribution xmlns="http://cloudfront.amazonaws.com/doc/2012-03-15/">
   <Id>the distribution ID</Id>
   <Status>InProgress</Status>
   <LastModifiedTime>2009-11-19T19:37:58Z</LastModifiedTime>
   <DomainName>d60EXAMPLEfxaaqy9.cloudfront.net</DomainName>
   <DistributionConfig xmlns="http://cloudfront.amazonaws.com/doc/2012-03-15/">

      <CustomOrigin>
         <DNSName>www.example.com</DNSName>
         <HTTPPort>80</HTTPPort>
         <HTTPSPort>443</HTTPSPort>
         <OriginProtocolPolicy>match-viewer</OriginProtocolPolicy>
      </CustomOrigin>
      <CallerReference>20091130090000</CallerReference>
      <CNAME>beagles.com</CNAME>
      <Comment>My comments</Comment>
      <Enabled>true</Enabled>
      <Logging>
         <Bucket>myawslogbucket.s3.amazonaws.com</Bucket>
         <Prefix>myprefix/</Prefix>
      </Logging>
   </DistributionConfig>
</Distribution>
```

After creating the distribution, it might take a few minutes for the distribution to deploy. You can monitor the progress by polling for status as described in the section, Monitor Your CloudFront Distribution to Check Its Status (p. 46).

## Monitor Your CloudFront Distribution to Check Its Status

To get status information about a distribution, you send a GET request on the 2012-03-15/distribution/*Distribution ID* resource. Your request and the response might look like the following examples.

**Example Request**

```
GET /2012-03-15/distribution/<distribution ID> HTTP/1.1
Host: cloudfront.amazonaws.com
Authorization: [AWS authentication string]
Date: [time stamp]
[Other required headers]
```

**Example Response**

```
200 OK
ETag: [ETag value to use later when doing a PUT or DELETE.]
x-amz-request-id: request_id

<Distribution xmlns="http://cloudfront.amazonaws.com/doc/</">
   <Id>the distribution ID</Id>
   <Status>Deployed</Status>
   <LastModifiedTime>2009-11-19T19:37:58Z</LastModifiedTime>
   <DomainName>d60EXAMPLEfxaaqy9.cloudfront.net</DomainName>
   <DistributionConfig xmlns="http://cloudfront.amazonaws.com/doc/2012-03-15/"

      <CustomOrigin>
          <DNSName>www.example.com</DNSName>
          <HTTPPort>80</HTTPPort>
          <OriginProtocolPolicy>http-only</OriginProtocolPolicy>
      </CustomOrigin>
      <CallerReference>20091130090000</CallerReference>
      <CNAME>beagles.com</CNAME>
      <Comment>My comments</Comment>
      <Enabled>true</Enabled>
      <Logging>
         <Bucket>myawslogbucket.s3.amazonaws.com</Bucket>
         <Prefix>myprefix/</Prefix>
      </Logging>
   </DistributionConfig>
</Distribution>
```

For more information on the ETag element, see Updating a Distribution's Configuration (p. 53).

When `Enabled` becomes `true` and `Status` changes from `InProgress` to `Deployed`, your distribution is ready. Also, as shown in the preceding example, the automatically generated domain name for your distribution is visible. If you chose to use a CNAME, it is also visible. It generally takes less than 15 minutes for a distribution to deploy.

You are ready to reference your content with your new Amazon CloudFront URL or CNAME.

# Reference Content in Your Web Pages or Applications

After you've created your distribution, Amazon CloudFront knows where your origin server is, and you know the domain name associated with the distribution. You can create a link to that domain name and CloudFront will serve it to your web page or application.

> **Note**
>
> You must wait until the distribution's state is Deployed before testing your links.

**To link to your object**

1.  Copy the following HTML into a new file and replace the domain name with your distribution's domain name and object name with the name of your content.

```
<html>
<head>My CloudFront Test</head>
<body>
<p>My text content goes here.</p>
<p><img src="http://<domain name>/<object name>" alt="my test image"/>
</body>
</html>
```

For example, if your domain name was `d604721EXAMPLE9.cloudfront.net` and your object was `image.jpg`, the URL for the link would be:
`http://d604721EXAMPLE9.cloudfront.net/image.jpg`.
If your object is in a folder on your origin server, then the folder must also be included in the URL. For example, if image.jpg was located in your origin images folder, then the URL would be: http://d604721EXAMPLE9.cloudfront.net/images/image.jpg.

2.   Save the changes to your file.

3.   Open your web page in a browser to ensure you can see your image.

The browser returns your page with the embedded image file, served from the edge location that CloudFront determined was appropriate to serve the object.

Congratulations! You've used CloudFront to serve an image from a custom origin.

# Requirements and Recommendations for Using Amazon EC2 and Other Custom Origins

Follow these guidelines for using Amazon EC2 instances and other custom origins with CloudFront.

*   Host and serve the same content on all servers.
*   Log the `X-Amz-Cf-Id` header entries on all servers; CloudFront requires this information for debugging.
*   Restrict access requests to the HTTP and HTTPS ports that your custom origin listens on.
*   Synchronize the clocks of all servers in your implementation.
*   Use redundant servers to handle failures.
*   For information about request and response behavior and about supported HTTP status codes, see Request and Response Behavior, and Supported HTTP Status Codes (p. 63).

If you use Amazon Elastic Compute Cloud for your custom origins, we recommend that you do the following:

*   Use an Amazon Machine Image that automatically installs the software for a web server. For more information, see the Amazon EC2 documentation.

*   Use an Elastic Load Balancing load balancer to handle traffic across multiple Amazon EC2 instances and to isolate your application from changes to Amazon EC2 instances. For example, if you use a load balancer, you can add and delete Amazon EC2 instances without changing your application. For more information, see the Elastic Load Balancing documentation.

*   Create a CloudFront distribution with a custom origin, and use the URL of the load balancer for the origin DNS name. For more information, see Creating a Distribution with a Custom Origin (p. 41).

# Using CNAMEs

**Topics**

## What Is a CNAME?

A CNAME record is an entry in a DNS table that lets you alias one fully qualified domain name to another. For example, if you owned `www.beagles.com` and `beagles.dogs.com`, you could create the following CNAME record that would specify `www.beagles.com` as an alias for `beagles.dogs.com`.

```
www.beagles.com CNAME beagles.dogs.com
```

A CNAME is useful because it lets you choose a domain name for your links instead of the domain name CloudFront provides you. Download and streaming distributions support CNAMES.

> ⚠️ **Important**
>
> You must be the owner of any domain name that you specify as an alias of your distribution's domain name.

You create a distribution, and CloudFront returns `d111111abcdef8.cloudfront.net` as the distribution's domain name. The link you would create to your object called `images/image.jpg` would be `http://d111111abcdef8.cloudfront.net/images/image.jpg`.

However, you want your links to use `example.com` instead of the `cloudfront.net` domain name that CloudFront provided. So you want the link to `images/image.jpg` to be `http://example.com/images/image.jpg` instead of `http://d111111abcdef8.cloudfront.net/images/image.jpg`.

> 📋 **Note**
>
> CloudFront doesn't support CNAMEs with HTTPS. If content is requested over HTTPS using CNAMEs, your end users' browsers will display the warning: *This page contains both secure and non-secure items.* To prevent this message from appearing, don't use CNAMEs with CloudFront HTTPS distributions.

## Setting Up a CNAME Alias

The following figure and table describe the process for setting up a CNAME alias so you can use your own domain name in your links instead of your distribution's CloudFront domain name.

**Process for Setting Up a CNAME Alias**

| | |
|---|---|
| 1 | Update your distribution's configuration to include the CNAME alias (using the `CNAME` element in the `DistributionConfig` (or `StreamingDistributionConfig`) object). |
| | If you're adding another CNAME alias to a distribution that already has one, see Multiple CNAME Aliases (p. 51). |
| | For information about distribution configuration and updating a distribution, see Updating a Distribution's Configuration (p. 53), and go to DistributionConfig Complex Type in the *Amazon CloudFront API Reference*. For more information about streaming distributions, go to StreamingDistributionConfig Complex Type in the *Amazon CloudFront API Reference*. |
| 2 | Confirm your update in task 1 is fully deployed. |
| | The update is fully deployed after the distribution's status returns to `Deployed`. If you don't wait for the update to be deployed, the links you create in the following steps might not work. For information about getting a distribution's status, go to GET Distribution Config in the *Amazon CloudFront API Reference*. For more information about streaming distributions, go to GET Streaming Distribution Config in the *Amazon CloudFront API Reference*. |
| 3 | Create a CNAME record in the DNS system to establish the alias between your domain name and the CloudFront domain name for your distribution. |
| | The exact procedure for configuring DNS depends on your DNS server or DNS provider and is beyond the scope of this document. |
| | ⚠ **Important** |
| | If you already have an existing CNAME record for your domain name, make sure you update *that* record or replace it with a new one that points to your distribution's domain name. |
| | Also, make sure your CNAME record points to your distribution's domain name, and *not* your origin server. |

| 4 | Confirm in the DNS system that the CNAME record points to your distribution's domain name. Use a DNS tool like dig (for information about dig, go to http://www.kloth.net/services/dig.php). The following shows an example dig request on a domain name called images.yourdomain.com, and the relevant part of the response. |
|---|---|
| | ```
[prompt]> dig images.yourdomain.com

; <<>> DiG 9.3.3rc2 <<>> images.yourdomain.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15917
;; flags: qr rd ra; QUERY: 1, ANSWER: 9, AUTHORITY: 2, ADDITIONAL: 0

;; QUESTION SECTION:
;images.yourdomain.com.     IN    A

;; ANSWER SECTION:
images.yourdomain.com. 10800 IN  CNAME  d111111abcdef8.cloudfront.net.
...
...
``` |
| | The line in the Answer Section shows the CNAME alias between the domain name images.yourdomain.com and the CloudFront distribution domain name d111111abcdef8.cloudfront.net. The CNAME record for your domain name is set up correctly if the name on the right side of CNAME in that line is your CloudFront distribution's domain name. If it's your Amazon S3 origin server bucket or some other domain name, then the CNAME record is set up incorrectly. In that case, go back to task 3 and correct the CNAME record to point to your distribution's domain name. |
| 5 | Test the CNAME alias. Create some test links that use your domain name in the URL instead of the distribution's CloudFront domain name. Test those links to confirm your content is being served correctly with the CNAME alias. |
| 6 | Go live with the CNAME alias. Substitute your domain name for the distribution's CloudFront domain name in your live links to objects. |

# Multiple CNAME Aliases

You can use more than one CNAME alias with a distribution. For example, you could have `alias1.example.com` and `alias2.example.com` both associated with your distribution's domain name. You can have up to 10 CNAME aliases per distribution. You can associate a particular CNAME alias with only one distribution.

⚠️ **Important**

When adding an additional CNAME alias to a distribution that already has one, make sure to include the original CNAME alias in the `DistributionConfig` object. Otherwise, your update erases the original CNAME alias and just adds the new one. This is because the process of updating a distribution's configuration replaces the entire configuration object; it doesn't add new items to it.

# No CNAME vs. an Empty CNAME

In a distribution's configuration, the lack of a CNAME element is not equivalent to an empty CNAME element (<CNAME/>). If you don't want the distribution to have a CNAME, then don't include a CNAME element when you create the distribution or update its configuration. If you do, CloudFront returns a MalformedXML error (for more information about errors, go to Errors in the *Amazon CloudFront API Reference*). The following table shows a correct and incorrect configuration object for a distribution with no CNAMEs. For more information about the configuration object, go to DistributionConfig Complex Type in the *Amazon CloudFront API Reference*.

| Correct | ```
<?xml version="1.0" encoding="UTF-8"?>
<DistributionConfig xmlns="http://cloudfront.amazonaws.com/doc/2012-
03-15/">
    <S3Origin>
        <DNSName>myawsbucket.s3.amazonaws.com</DNSName>
    </S3Origin>
    <CallerReference>20120229090000</CallerReference>
    <Comment>My comments</Comment>
    <Enabled>true</Enabled>
</DistributionConfig>
``` |
|---|---|
| Incorrect | ```
<?xml version="1.0" encoding="UTF-8"?>
<DistributionConfig xmlns="http://cloudfront.amazonaws.com/doc/2012-
03-15/">
    <S3Origin>
        <DNSName>myawsbucket.s3.amazonaws.com</DNSName>
    </S3Origin>
    <CallerReference>20120229090000</CallerReference>
    <CNAME/>
    <Comment>My comments</Comment>
    <Enabled>true</Enabled>
</DistributionConfig>
``` |

To remove a CNAME from a distribution, remove the corresponding CNAME element and update the configuration. For information about updating a distribution's configuration, see .

# Updating a Distribution's Configuration

You can update a distribution's configuration at any time. This section describes the process you must use (regardless of the type of distribution: download or streaming). The only parts of the distribution's configuration you can update are the CNAMEs, the comments, minimum TTL, and whether the distribution is enabled. For more information about the contents of a download distribution's configuration, go to DistributionConfig Complex Type in the *Amazon CloudFront API Reference*. For more information about streaming distributions, go to StreamingDistributionConfig Complex Type in the *Amazon CloudFront API Reference*.

> ⚠️ **Important**
>
> When you update a distribution's configuration, you replace the entire configuration object with a new one; you don't add to the existing configuration. This distinction is important to remember when adding an additional CNAME alias to a distribution that already has one. Make sure to include the original CNAME alias in the `DistributionConfig` object, or else your update will erase the original CNAME alias and just add the new one.

When you try to update a distribution's configuration, another authorized person in your company could be trying to do the same thing. The following figure and table describe how to update the distribution to avoid any conflicts.



**Process for Updating a Distribution's Configuration**

| 1 | You send a request to retrieve the distribution's current configuration information. |
|---|---|
|   | You can get either the distribution's general information (go to GET Distribution, or specifically its configuration (go to GET Distribution Config). |
| 2 | CloudFront responds with the distribution's current information (in the form of an XML document) and the HTTP `ETag` header associated with the distribution's current information (e.g., E2QWRUHEXAMPLE). The same `ETag` is returned whether you requested the general distribution information or specifically the configuration. |
|   | If you requested the configuration, you can just update that same XML document you received with the changes you want to make to the distribution's configuration. |
| 3 | You send a request to update the distribution's configuration information. In the request, you include the updated XML document, and the HTTP `If-Match` header set to the value of the `ETag` header you received in task 2. |
|   | For information about updating a download distribution's configuration object, go to PUT Distribution Config in the *Amazon CloudFront API Reference*. For information about updating a streaming distribution's configuration object, go to PUT Streaming Distribution Config in the *Amazon CloudFront API Reference*. |

| 4 | AWS confirms that the version number you sent in the `If-Match` header matches the current version of the configuration information. |
|---|---|
| | If it does, your request is processed. |
| | If it does not, this means that someone else has modified the configuration since you retrieved the information earlier. In this case, CloudFront returns a `PreconditionFailed` error (HTTP status code 412). You should then repeat tasks 1-3. In task 2 you'll get a different value for the `ETag` header than you did the first time, and you must provide that new value in task 3. |

# Deleting a Distribution

You can have up to 100 distributions of each type (download and streaming) in the Amazon CloudFront system. However, you might find that you no longer want to use a particular distribution. The following figure and table describe the process you must use to delete a distribution.



**Process for Deleting a Distribution**

| 1 | You disable the distribution and get the value of the ETag header in the response to your PUT request. |
|---|---|
| | Specifically, you update the distribution's configuration to have the `Enabled` value set to `false`. For information about updating a distribution's configuration, see Updating a Distribution's Configuration (p. 53). |
| 2 | You confirm that the distribution is disabled. |
| | This task is optional; if you try to delete the distribution before it is fully disabled, CloudFront returns a `DistributionNotDisabled` error (HTTP status code 409). |
| | To confirm the distribution is disabled, you fetch the distribution's information and confirm the value of the `Status` is `Deployed` and not `InProgress` (`Deployed` indicates your request to disable the distribution has propagated throughout the CloudFront system). For information about fetching a download distribution's information, go to GET Distribution in the *Amazon CloudFront API Reference*. |
| | After the distribution is disabled, any end-user requests that use the domain name associated with the distribution will fail. |
| 3 | You send a request to delete the distribution and include the HTTP `If-Match` header set to the value of the `ETag` header you received in task 1. |
| | For more information, go to DELETE Distribution in the *Amazon CloudFront API Reference*. |

| 4 | AWS confirms that the version number you sent in the `If-Match` header matches the current version of the configuration information. |
|---|---|
| | If it does, your request is processed. |
| | If it does not, this means that someone else has modified the distribution since you disabled it (and possibly re-enabled it). In this case, CloudFront returns a `PreconditionFailed` error (HTTP status code 412). You should then repeat tasks 1-3. In task 1 you'll get a different value for the `ETag` header than you did the first time, and you must provide that new value in task 3. |

# Streaming Media on Demand

**Topics**

The basic CloudFront distribution is a download distribution that serves HTTP content. You might want to also serve streamed media using a streaming protocol. This section describes how to use CloudFront to stream on-demand media files.

With CloudFront, *streaming* means that the end user receives and uses (for example, views) media simultaneously. A streaming server streams the data over a special protocol to a media player that plays the media as it receives it. When the end user is done viewing the video (for example), the video isn't stored locally on the end user's system.

Streaming distributions differ from download distributions in the following ways:

| Download | Streaming |
|---|---|
| Content distributed over HTTP | Content distributed over RTMP |
| Generally, viewers render content after it downloads completely | Content used as it downloads |

**Note**

CloudFront currently only supports Amazon S3 origin servers for streaming distributions. Custom origins are not supported for streaming distributions. In addition, you must be using API version 2009-12-01 or later.

# Streaming Server and Media Player

CloudFront uses Adobe Flash Media Server 3.5 to stream on-demand content with Adobe's Real-Time Messaging Protocol (RTMP). CloudFront accepts RTMP requests over port 1935 and port 80.

CloudFront supports the following variants of the RTMP protocol:

- **RTMP—**Adobe's Real-Time Message Protocol
- **RTMPT—**Adobe streaming tunneled over HTTP
- **RTMPE—**Adobe encrypted
- **RTMPTE—**Adobe encrypted tunneled over HTTP

Which protocol you use is up to you and depends on your own needs. For a basic summary of RTMP and the file formats that Adobe Flash Media Server supports, go to Overview of Streaming with Flash Media Server 3 on the Adobe website. This includes information about the supported codecs and containers.

There are resources available on the Internet to help you determine the bit rate to use for your Flash files. For an example, go to Flash video (FLV) bitrate calculator on the Adobe website.

CloudFront supports all the features in Adobe Flash Media Server 3.5 related to *dynamic streaming*, which is the ability to switch between different quality streams during playback. For more information, go to Dynamic streaming in Flash Media Server 3.5: Part 1 on the Adobe website.

To serve streamed content, you need to provide your end users with a media player. You can write your own player using Adobe Flash. For more information, go to http://www.adobe.com/products/flashplayer/. You can also use an existing player, such as JW Player. For more information, go to http://www.longtailvideo.com/.

# Layout of Buckets and Distributions

To stream content, you need to provide two types of files to your end users: your media files and a media player. In this topic, we assume that your media files and your media player are stored in different buckets in Amazon S3, and are served to end users through CloudFront. Making the media player available to end users through Amazon S3 and CloudFront is optional. You can also refer end users to a media player that is stored elsewhere.

If you're using CloudFront to serve both the media player and the media files, you need to use two types of distributions: a download distribution for the media player, and a *streaming distribution* for the media files. Download distributions serve files over HTTP, whereas streaming distributions stream media files over RTMP (or a variant of RTMP).

The following diagram shows the general layout of your CloudFront streaming setup.

| | |
|---|---|
| **1** | Your media player bucket holds the media player and is the origin server for a regular HTTP distribution. In this example, the distribution's domain name is d1234.cloudfront.net. |
| **2** | Your streaming bucket holds your media files and is the origin server for a streaming distribution. In this example, the distribution's domain name is s5678.cloudfront.net. |

Your site serves a cached copy of the media player to each end user (through the d1234.cloudfront.net domain). The media player then accesses cached copies of your media files (through the s5678.cloudfront.net domain).

# Overall Process to Stream Content

The following table describes the process for configuring on-demand streaming. If you are using JW Player, Flowplayer, or Adobe Flash Builder for your media player, see the applicable tutorial instead:

- JW Player
- Flowplayer
- Adobe Flash Builder

**Process for Streaming Content**

| 1 | Choose and configure a media player to play your media files. For help, refer to the documentation for the media player. |
|---|---|

| 2 | Upload the files for your media player to the Amazon S3 bucket that you created for the media player, and make the files (not the bucket) publicly readable. |
|---|---|
| 3 | Create a distribution for your media player (or you can use an existing distribution). You can use the CloudFront control API or your favorite CloudFront tool. For a list of tools, go to Amazon CloudFront Developer Tools. <br><br> For information on creating a download distribution with the control API, go to POST Distribution in the *Amazon CloudFront API Reference*. |
| 4 | Upload your media files to the Amazon S3 bucket that you created for the media files, and make the content (not the bucket) publicly readable. <br><br> ⚠️ **Important** <br><br> Media files in a Flash Video container must include the .flv filename extension, or the media will not stream. <br><br> You can put media player files and media files in the same bucket. |
| 5 | Use the CloudFront API (or your favorite CloudFront tool) to create a streaming distribution. For more information, see Creating a Streaming Distribution (p. 59). |
| 6 | Configure your media player. For more information, see Configuring the Media Player (p. 60). |

If you have trouble getting your content to play, see Troubleshooting (p. 62).

# Creating a Streaming Distribution

Creating a streaming distribution is almost identical to creating a download distribution. The easiest way is to use the AWS Management Console at http://aws.amazon.com/console.

📑 **Note**

CloudFront currently only supports Amazon S3 origin servers for streaming distributions. Custom origins are not supported for streaming distributions.

**To create a streaming distribution with the AWS Management Console**

1. Sign in to the AWS Management Console and open the Amazon CloudFront console at https://console.aws.amazon.com/cloudfront/.
2. Click **Create Distribution**.
   The **Create Distribution** dialog box opens.
3. Select the **Streaming** button to specify a streaming distribution.
4. In the **Origin** list, select the bucket you want as your origin server.
5. Leave the remaining items as they are and click **Create**.
   Your new streaming distribution is created and appears in the list of distributions on the page. You can select the check box for the distribution to view its information at the bottom of the page.

The CloudFront control API also provides a set of actions for creating and managing your streaming distributions. The actions are parallel to those for creating and managing download distributions. The

main difference is that the resource is a `streaming-distribution` instead of a `distribution`. For information about the API actions for streaming distributions, go to the Amazon CloudFront API Reference.

**To create a streaming distribution with the control API**

1. Send a CloudFront control API request that looks similar to the following example.

```
POST /2012-03-15/streaming-distribution HTTP/1.1
[Required headers]

<?xml version="1.0" encoding="UTF-8"?>
<StreamingDistributionConfig xmlns="http://cloudfront.amazonaws.com/doc/2012-
03-15/">
   <S3Origin>
       <DNSName>mystreamingbucket.s3.amazonaws.com</DNSName>
   </S3Origin>
   <CallerReference>20120229090000</CallerReference>
   <Comment>My comments</Comment>
   <Enabled>true</Enabled>
</StreamingDistributionConfig>
```

2. You will receive a response that looks similar to the following example.

```
201 Created
Location: https://cloudfront.amazonaws.com/2012-03-15/streaming-distribu
tion/EGTXBD79EXAMPLE
x-amz-request-id: request_id

<?xml version="1.0" encoding="UTF-8"?>
<StreamingDistribution xmlns="http://cloudfront.amazonaws.com/doc/2012-03-
15/">
    <Id>EGTXBD79EXAMPLE</Id>
    <Status>InProgress</Status>
    <LastModifiedTime>2009-11-19T19:37:58Z</LastModifiedTime>
    <DomainName>s5c39gqb8ow64r.cloudfront.net</DomainName>
    <StreamingDistributionConfig>
        <S3Origin>
            <DNSName>mystreamingbucket.s3.amazonaws.com</DNSName>
        </S3Origin>
        <CallerReference>20120229090000</CallerReference>
        <Comment>My comments</Comment>
        <Enabled>true</Enabled>
    </StreamingDistributionConfig>
</StreamingDistribution>
```

3. Store the `Location` header from the response, which contains the URI for your newly created streaming distribution, the ID for the distribution, and the domain name.

# Configuring the Media Player

To play a media file, you need to configure the media player with the correct path to the file. How you configure the media depends on which media player you're using and how you're using it.

When you configure the media player, the path you specify to the media file must contain the characters `cfx/st` immediately after the domain name. For example, `rtmp://s5c39gqb8ow64r.cloudfront.net/cfx/st/mediafile.flv`.

> **Note**
>
> CloudFront follows Adobe's FMS naming requirements. Different players have their own rules about how to specify streams. The above example is for JW Player. Check your player's documentation. For example, Adobe's Flash Media Server does not allow the `.flv` extension to be present on the play path. Many players try and hide this fact and actually remove the `.flv` extension for you.

Your media player might ask for the path separate from the file name. For example, with JW Player's wizard, you specify a *streamer* and *file* variable:

- **streamer—**`rtmp://s5c39gqb8ow64r.cloudfront.net/cfx/st` (with no trailing slash)
- **file—** `mediafile.flv`

If you've stored the media files in a directory in your bucket (for example, `videos/mediafile.flv`), then the variables for JW Player would be:

- **streamer—**`rtmp://s5c39gqb8ow64r.cloudfront.net/cfx/st` (with no trailing slash)
- **file—** `videos/mediafile.flv`

To use the JW Player wizard, go to http://www.longtailvideo.com/support/jw-player-setup-wizard?example=204.

## MPEG Files

To serve MP3 audio files or H.264/MPEG-4 video files, you might need to prefix the file name with `mp3:` or `mp4:`. Some media players can be configured to add the prefix automatically. The media player might also require you to specify the file name without the file extension (for example, `magicvideo` instead of `magicvideo.mp4`).

# Restricting Access Using Crossdomain.xml

The Adobe Flash Media Server `crossdomain.xml` file specifies which domains can access media files in a particular domain. CloudFront supplies a default file that allows all domains to access the media files in your streaming distribution, and you cannot change this behavior. If you include a more restrictive `crossdomain.xml` file in your S3 bucket, CloudFront ignores it.

# Error Codes

The following table lists the error codes that CloudFront can send to your playback media player. The errors are part of the string returned with `Event.info.application.message` or `Event.info.description`.

| Error | Description |
| --- | --- |
| DistributionNotFound | The distribution is not found |

| Error | Description |
|-------|-------------|
| `DistributionTypeMismatch` | The distribution is not a streaming distribution |
| `InvalidInstance` | The instance is invalid |
| `InvalidURI` | The URI is invalid |

# Troubleshooting

If you're having trouble getting your media files to play, check the items in the following table.

| Item to Check | Description |
|---------------|-------------|
| Separate distributions for the media player files and media files | The media player must be served by a regular HTTP distribution (for example, domain name d111111abcdef8.cloudfront.net), whereas the media files must be served by a streaming distribution (for example, domain name s5c39gqb8ow64r.cloudfront.net). Make sure you're not using the same distribution for both. |
| `/cfx/st` in the file path | Make sure to include `/cfx/st` in the file's path (for more information, see Configuring the Media Player (p. 60)). You don't need to include `/cfx/st` in the path to the object in the Amazon S3 bucket. |
| Problems playing MPEG-4 files | Some media players require `mp4:` before the file name. Some might also require you to exclude the `.mp4` extension. For more information, see MPEG Files (p. 61). |
| Port 1935 on your firewall | Adobe Flash Media Server uses port 1935 for RTMP. Make sure your firewall has this port open. If it doesn't, the typical message returned is "Unable to play video." You can also switch to RTMPT to tunnel over HTTP using port 80. |
| Adobe Flash Player messaging | By default, the Adobe Flash Player won't display a message if the video file it's trying to play is missing. Instead, it waits for the file to show up. You might want to change this behavior to give your end users a better experience.<br><br>To instead have the player send a message if the video is missing, use `play("vid",0,-1)` instead of `play("vid")`. |

# Request and Response Behavior, and Supported HTTP Status Codes

See the applicable section:

## Request and Response Behavior for Amazon S3 Origins

**Topics**

### How CloudFront Processes and Forwards Requests to Your Amazon S3 Origin Server

For information about how CloudFront processes end-user requests and forwards the requests to your Amazon S3 origin, see the applicable topic:

### HTTP Methods

CloudFront accepts only `GET` and `HEAD` requests from end users.

## Query Strings

CloudFront does not forward query string parameters to your Amazon S3 origin. For more information, see How CloudFront Processes and Logs Query String Parameters (p. 23).

## Protocols

CloudFront forwards HTTP or HTTPS requests to the origin server based on the protocol of the request that the end user sends to CloudFront, either HTTP or HTTPS.

## IP Addresses

The IP address that CloudFront forwards to Amazon S3 is the IP address of a CloudFront server, not the IP address of the end user's computer.

## Caching Duration and Minimum TTL

To control whether CloudFront caches an object and for how long, we recommend that you use the `Cache-Control` header with the `max-age=<seconds>` directive. CloudFront caches the object for the specified number of seconds. (The minimum value is 0 seconds.) After the specified amount of time has elapsed, CloudFront forwards the next request for the object to the origin to verify that the cache contains the latest version of the object:

- If CloudFront already has the latest version, the origin returns only a 304 status code (not modified).
- If CloudFront does not have the latest version, the origin returns a 200 status code (OK) and the latest version of the object.

If an object is seldom requested, CloudFront may evict the object from the cache regardless of the value of the max-age directive.

Although you can also use the `Expires` header field to control object caching, we recommend that you use the `Cache-Control` header field instead. If you specify values both for `Cache-Control max-age` and for `Expires`, CloudFront uses only the value of `max-age` to control object caching.

You can specify the minimum amount of time that CloudFront keeps an object in the cache before forwarding another request to the origin. This value, the minimum TTL, applies to all of the objects in a distribution. An object stays in cache for the longer of the following durations:

- The value specified by the `max-age` directive.
- The minimum TTL for the distribution.

If the minimum TTL is 0, then CloudFront additionally supports the `no-cache`, `no-store`, and `private` directives in the `Cache-Control` header on each object. For more information about specifying the minimum TTL for a distribution, in the *Amazon CloudFront API Reference*, see the `CachingBehavior` element in DistributionConfig Complex Type.

For information about additional options, see "Section 14.9 Cache Control" in *Hypertext Transfer Protocol -- HTTP/1.1* at http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.9.

For more information, see Specifying When an Object Expires (p. 25).

# How CloudFront Processes Responses from Your Amazon S3 Origin Server

The maximum size of a response body that CloudFront will return to the end user is 20 GB. This includes chunked transfer responses that don't specify the `Content-Length` header value.

# Request and Response Behavior, and Supported HTTP Status Codes for Custom Origins

**Topics**

## How CloudFront Processes and Forwards Requests to Your Custom Origin Server

For information about how CloudFront processes end-user requests and forwards the requests to your custom origin, see the applicable topic:

### HTTP Methods

CloudFront accepts only `GET` and `HEAD` requests from end users.

### Query Strings

CloudFront does not forward query string parameters to your origin. For more information, see How CloudFront Processes and Logs Query String Parameters (p. 23).

# HTTP Version

CloudFront forwards requests to your custom origin using HTTP/1.0, but supports most of the HTTP 1.1 specification. To enhance performance, we recommend that you include the `Keep-Alive` header in end-user requests.

# Encryption

CloudFront forwards HTTPS requests to the origin server using the RC4-MD5 cipher.

# Protocols

CloudFront forwards HTTP or HTTPS requests to the origin server based on the following:

- The protocol of the request that the end user sends to CloudFront, either HTTP or HTTPS
- The setting of the `OriginProtocolPolicy` element in the `DistributionConfig` complex type, either `http-only` or `match-viewer`.

You can only specify the origin protocol policy when you create a distribution, using either the Create Distribution Wizard or the `POST Distribution` API action. After you create a distribution, you can't change the setting. If you aren't sure which protocol to use, we recommend that you specify `http-only`. For information about how to specify the protocol policy when you create a distribution using the Create Distribution Wizard, see Creating a Distribution with a Custom Origin (p. 41). For information about how to specify the origin protocol policy when you create a distribution using the `POST Distribution` action, see POST Distribution in the Amazon CloudFront API Reference.

If you specify `http-only` for the `OriginProtocolPolicy` element, CloudFront forwards requests to the origin server using only the HTTP protocol, regardless of the protocol in the end-user request.

If you specify `match-viewer` for the `OriginProtocolPolicy` element, CloudFront forwards requests to the origin server using the protocol in the end-user request.

> **Caution**
>
> If the end-user request uses the HTTPS protocol, and if the origin server returns an invalid certificate or a self-signed certificate, CloudFront drops the TCP connection.

# Client Authentication

Do not configure your origin server to request client authentication. When CloudFront forwards an end-user request to your origin server over HTTPS, CloudFront does not present a certificate.

# Removed Header Fields

CloudFront removes hop-by-hop header fields such as the `Authorization` and `Connection` fields before forwarding requests to your origin.

# IP Addresses

The IP address that CloudFront forwards to the origin server is the IP addresses of a CloudFront server, not the IP address of the end user's computer.

# Compression

- CloudFront forwards requests that have the `Accept-Encoding` field values `"identity"` and `"gzip, deflate"`. CloudFront accepts `"deflate, gzip"` and reorders the values to `"gzip, deflate"`.
- By default, IIS does not serve compressed content for requests that come through proxy servers such as CloudFront. If you're using IIS and if you configured IIS to compress content by using the `httpCompression` element, change the values of the `noCompressionForHttp10` and `noCompressionForProxies` attributes to false.

# Cookies

CloudFront does not forward `Cookie` header fields to your origin. In addition, if the origin server includes `Set-Cookie` header fields, CloudFront removes these headers before caching the objects at the edge location and before serving the objects to the end user.

# Caching Duration and Minimum TTL

To control whether CloudFront caches an object and for how long, we recommend that you use the `Cache-Control` header with the `max-age=<seconds>` directive. CloudFront caches the object for the specified number of seconds. (The minimum value is 0 seconds.) After the specified amount of time has elapsed, CloudFront forwards the next request for the object to the origin to verify that the cache contains the latest version of the object:

- If CloudFront already has the latest version, the origin returns only a 304 status code (not modified).
- If CloudFront does not have the latest version, the origin returns a 200 status code (OK) and the latest version of the object.

If an object is seldom requested, CloudFront may evict the object from the cache regardless of the value of the max-age directive.

Although you can also use the `Expires` header field to control object caching, we recommend that you use the `Cache-Control` header field instead. If you specify values both for `Cache-Control max-age` and for `Expires`, CloudFront uses only the value of `max-age` to control object caching.

You can specify the minimum amount of time that CloudFront keeps an object in the cache before forwarding another request to the origin. This value, the minimum TTL, applies to all of the objects in a distribution. An object stays in cache for the longer of the following durations:

- The value specified by the `max-age` directive.
- The minimum TTL for the distribution.

If the minimum TTL is 0, then CloudFront additionally supports the `no-cache`, `no-store`, and `private` directives in the `Cache-Control` header on each object. For more information about specifying the minimum TTL for a distribution, in the *Amazon CloudFront API Reference*, see the `CachingBehavior` element in DistributionConfig Complex Type.

For information about additional options, see "Section 14.9 Cache Control" in *Hypertext Transfer Protocol -- HTTP/1.1* at http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.9.

For more information, see Specifying When an Object Expires (p. 25).

# How CloudFront Processes Responses from Your Custom Origin Server

For information about how CloudFront processes responses from custom origin servers, see the applicable topic:

## Maximum File Size

The maximum size of a response body that CloudFront will return to the end user is 20 GB. This includes chunked transfer responses that don't specify the `Content-Length` header value.

## Caching

- Ensure that the origin server sets valid and accurate values for the `Date` and `Last-Modified` header fields.
- If requests from end users include the `If-Match` or `If-None-Match` request header fields, set the `ETag` response header field. If you do not specify an `ETag` value, CloudFront ignores subsequent `If-Match` or `If-None-Match` headers.

## Content Negotiation

The only acceptable value for the `Vary` header is `Accept-Encoding`. CloudFront ignores other values.

## Redirects

If you change the location of an object on the origin server, you can configure your web server to redirect requests to the new location. After you configure the redirect, the first time an end user submits a request for the object, CloudFront Front sends the request to the origin, and the origin responds with a redirect (for example, `302 Moved Temporarily`). CloudFront caches the redirect and returns it to the end user. CloudFront does not follow the redirect.

You can configure your web server to redirect requests to one of the following locations:

- The new URL of the object on the origin server. When the end user follows the redirect to the new URL, the end user bypasses CloudFront and goes straight to the origin. As a result, we recommend that you not redirect requests to the new URL of the object on the origin.
- The new CloudFront URL for the object. When the end user submits the request that contains the new CloudFront URL, CloudFront gets the object from the new location on your origin, caches it at the edge location, and returns the object to the end user. Subsequent requests for the object will be served by the edge location. This avoids the latency and load associated with viewers requesting the object from the origin. However, every new request for the object will incur charges for two requests to CloudFront.

## Transfer Encoding

CloudFront supports only transfer encoding and requires the `Transfer-Encoding` header field value to be `"chunked"`.

# Supported HTTP Status Codes for Custom Origin Servers

If your custom origin server responds to a CloudFront request with any of the following status codes, CloudFront caches the code for five minutes and writes the results to the access logs.

| | |
|---|---|
| 204 | No Content |
| 305 | Use Proxy |
| 400 | Bad Request |
| 403 | Forbidden |
| 404 | Not Found |
| 405 | Method Not Allowed |
| 414 | Request-URI Too Large |
| 500 | Internal Service Error |
| 501 | Not Implemented |
| 502 | Bad Gateway |
| 503 | Service Unavailable |
| 504 | Gateway Time-out |

# Using a Signed URL to Serve Private Content

**Topics**

For many companies that distribute data via the Internet, it is important to restrict access to documents, business data, media streams, or content intended for users who have paid a fee. You can use CloudFront private distributions to restrict access to data in Amazon S3 buckets. This section describes how a private distribution is different from a public distribution, it describes how to create a private distribution, and it provides links to sample code you might find helpful when creating your signed URL.

> **Note**
>
> Private content is not supported for custom origins. You can use a signed URL to distribute content from a custom origin, but for CloudFront to access the origin, the origin must remain publicly accessible. For more information about custom origins, see The Origin Server (p. 38).

## Static Signed URLs vs. Dynamic Signed URLs

You can distribute private content with a static signed URL or a dynamic signed URL. You use a static signed URL when distributing private content to a known end user, such as distributing a business plan to an investor, or distributing training materials to employees. In this case, you create a signed URL and make the URL available to your end users as needed. You use a dynamic signed URL to distribute content on-the-fly to an end user for a limited purpose, such as distributing movie rentals or music downloads to customers on demand. In this case, your application generates the signed URL.

To integrate signed URL creation into your application for dynamic, on-the-fly signed URL generation, follow the procedures described in this section. However, to avoid coding, and yet distribute content to an end user for a limited purpose without dynamic signed URL creation, you can try creating a CloudFront private distribution using one of the third-party GUI tools listed in GUI Tools for Signature Generation (p. 112).

# Overview of Private Content

**Topics**

This section describes the differences between public and private content, and explains the structure of a private content distribution.

## Public vs. Private Content

You can think of your content as being either *public* or *private* (i.e., restricted). The following table gives a general description of what each means.

| Type of Content | General Description | General Implementation |
| --- | --- | --- |
| Public | You want anyone to be able to access your objects | You set the Amazon S3 Access Control List (ACL) on the objects in your bucket to give `read` permission to everyone. This means that end users can access the objects through either CloudFront or Amazon S3. |
| Private | You want to restrict who can access your objects (for example, to control access to a paid download) | You set the Amazon S3 ACL on your objects so that only you and CloudFront have `read` permission for the objects. This means that end user access to the objects can *only* be through CloudFront. You also produce signed URLs for the end users you want to give access to. |

A CloudFront private distribution is based on a *policy statement* that specifies any or all of the following constraints:

- A start date that specifies the date and time the signed URL will be valid

- An end date and time after which the signed URL will not be valid

- An IP address or range of IP addresses from which the signed URL can be used

![note icon] **Note**

> The two initial API versions for CloudFront (2008-06-30 and 2009-04-02) require your content to be public. As of version 2009-09-09 you can serve public content, private content, or both.

## How Private Distributions Are Different from SSL and HTTPS

The security provided by a CloudFront private distribution is different from SSL and HTTPS, which encrypt the data that is transferred over a network connection. A CloudFront private distribution encodes request parameters that are appended to the base URL. The encoded parameters contain a *policy statement* (with the restrictions described in Public vs. Private Content (p. 71)) and a *signature*, which authenticates that the policy was generated by a trusted signer and has not been tampered with. When authentication

succeeds, the encoded content of the policy statement authorizes the request for data in the distribution during the specified time period and from the specified IP address or addresses.

For an overview of the process of creating private content, see Two Parts to Serving Private Content (p. 72).

# Two Parts to Serving Private Content

The process of serving private content has two main parts described in the following diagram and table.



| | |
|---|---|
| **1** | Securing the content in your bucket so that end users *only* have access to the content through CloudFront. |
| | You remove all public access to the objects in your buckets, but still give CloudFront permission to fetch the objects. To do this, you create a *CloudFront origin access identity*, which is a virtual identity that you give `read` permission to. Granting that permission allows CloudFront to fetch content that is otherwise marked as private in your bucket. |
| **2** | Restricting end user access to cached content. |
| | You hand out special signed URLs to end users you want to give access to. The URLs contain restrictions limiting access to the private content. |

You can do both parts, or just part 1. If you just do part 1, you still use public URLs just as you do to serve public content. The URL remains publicly accessible, but it is accessible only through CloudFront (not through your Amazon S3 bucket) and you don't need to use a signed URL. If you do part 2, the URL is private and is not publicly accessible.

For information about how to create a private content distribution using a signed URL, see How to Serve Private Content Using a Signed URL (p. 73).

# How to Serve Private Content Using a Signed URL

**Topics**

This section describes how to create a distribution you can use to distribute private content via a signed URL.

## Private Content Process Overview

The following figure and table describe the process for restricting user access. The process is divided into two sections corresponding to the two parts of the process (as described in Two Parts to Serving Private Content (p. 72)).

The first section, the top row with the first three tasks shaded in blue, covers the tasks required to secure your content in Amazon S3 (that is, to make your content accessible only through CloudFront).

The second section, with tasks 4 through 6, covers the additional tasks required to create signed URLs, which allow you to restrict access to the content to users who have a signed URL.



**Process for Serving Private Content**

| 1 | Use the CloudFront API to create a *CloudFront origin access identity*. |
|---|---|
| | For more information, see Overview of the CloudFront Origin Access Identity (p. 75). |
| 2 | Use the Amazon S3 API or the Amazon S3 console of the AWS Management Console to update the ACL on your private objects. Give `read` permission to the CloudFront origin access identity by using its canonical user ID. Remove public-access permissions. |
| | For more information about setting the ACL, see Updating Amazon S3 Bucket Policies or ACLs on Your Private Content Buckets or Objects (p. 78). |
| 3 | Set up a private content distribution or streaming distribution (either create a new distribution or update an existing one). |
| | For more information, see Creating a Private Content Distribution (p. 77). |

| 4 | Use the **Accounts** link in the AWS Management Console to access the **Key Pairs** tab of the **Access Credentials** page. Create an RSA key pair and download the private key. You'll use this key to create a signed URL. |
| | For more information about creating your key pair, see Creating a Key Pair (p. 80). |
| 5 | Update your private content distribution or streaming distribution to specify that the distribution's URLs must be signed, and the accounts that can sign them. |
| | For more information, see Requiring Signed URLs (p. 81). |
| 6 | Create a signed URL to give to the authorized end user. For more information, see Signature Code, Examples, and Tools (p. 97). |

To get started with creating a private content distribution using a signed URL, see Securing Your Content in Amazon S3 (p. 75).

# Securing Your Content in Amazon S3

**Topics**

To use CloudFront private distributions, you secure your content in Amazon S3 by creating a CloudFront origin access identity and by setting the ACL on the objects or buckets in Amazon S3 to be accessible only by that identity. This section describes what an origin access identity is, shows you how to modify your distribution to include a CloudFront *origin access identity*, and how to authorize CloudFront access to data in Amazon S3.

## Overview of the CloudFront Origin Access Identity

A *CloudFront origin access identity* is a virtual identity that allows CloudFront to fetch content from an Amazon S3 bucket. You create a CloudFront origin access identity for your AWS account, attach the identity to your distribution, and then give that identity `read` permission (or `read and download` permission) to objects in Amazon S3. After you remove public access to the Amazon S3 bucket, the CloudFront distribution is now the only way to access objects in your bucket. Adding *signer* accounts to the distribution configuration allows access only to users who have signed URLs.

You can have up to 100 CloudFront origin access identities, and you can attach each to one or more distributions. One origin access identity is usually sufficient, even for multiple distributions.

The following example depicts three different distributions.



| | Distribution 1 is configured for public content. The object has an Amazon S3 ACL that grants everyone `read` permission. Anyone can access the contents of this distribution through Amazon S3 or through your CloudFront distribution. |
|---|---|

| | |
|---|---|
| **2** | Distribution 2 is configured to read private content with signed URLs. This distribution is attached to CloudFront origin access identity *A*. The object has an Amazon S3 ACL that grants `read` permission to the identity. The content in this distribution cannot be accessed by anyone who doesn't have the signed URL. |
| **3** | Distribution 3 is configured to read private content with public URLs. This distribution is also attached to CloudFront origin access identity A. The object has an Amazon S3 ACL that grants `read` permission to the identity. The content in this distribution is not private, but users can access it only through your CloudFront distribution (not through your Amazon S3 bucket). |

For information about creating an origin access identity, see Creating a CloudFront Origin Access Identity (p. 76).

# Creating a CloudFront Origin Access Identity

You can create a CloudFront origin access identity using a `POST` on the `2012-03-15/origin-access-identity/cloudfront` resource. You must provide a unique caller reference in the request, as you do when creating a distribution. You can optionally provide comments about the identity.

![Note icon] **Note**

> Currently, the AWS Management Console doesn't support creating an origin access identity or updating a distribution to serve private content.

**To create a CloudFront origin access identity for your distribution**

1. Send a CloudFront control API request that is similar to the following example.

```
POST /2012-03-15/origin-access-identity/cloudfront HTTP/1.1
[Required headers]

<?xml version="1.0" encoding="UTF-8"?>
<CloudFrontOriginAccessIdentityConfig xmlns="http://cloudfront.amazon
aws.com/doc/2012-03-15/">
  <CallerReference>20120229090000</CallerReference>
  <Comment>Your comments here</Comment>
</CloudFrontOriginAccessIdentityConfig>
```

2. You will receive a response that looks similar to the following example.

```
201 Created
Location: https://cloudfront.amazonaws.com/2012-03-15/origin-access-iden
tity/cloudfront/E74FTE3AEXAMPLE
x-amz-request-id: request_id

<?xml version="1.0" encoding="UTF-8"?>
<CloudFrontOriginAccessIdentity xmlns="http://cloudfront.amazon
aws.com/doc/2012-03-15/">
  <Id>E74FTE3AEXAMPLE</Id>
  <S3CanonicalUserId>
     cd13868f797c227fbea2830611a26fe0a21ba1b826ab4bed9b7771c9aEXAMPLE
  </S3CanonicalUserId>
```

```
    <CloudFrontOriginAccessIdentityConfig>
      <CallerReference>20120229090000</CallerReference>
      <Comment>Your comments here</Comment>
    </CloudFrontOriginAccessIdentityConfig>
</CloudFrontOriginAccessIdentity>
```

3. Record the `Id` and the `S3CanonicalUserId` for the new CloudFront origin access identity.

   You will use these values later in the process. The `Id` attaches the origin access ID to a distribution, and the `S3CanonicalUserId` identifies CloudFront in the Amazon S3 ACL on the object. For more information about origin access ID or the Canonical User ID, go to Actions on Origin Access Identities in the *Amazon CloudFront API Reference*.

> **Note**
>
> The CloudFront control API provides a set of actions for creating and managing your CloudFront origin access identities. The actions are parallel to those for creating and managing distributions. For more information about the actions, go to Actions on Origin Access Identities in the *Amazon CloudFront API Reference*.

Now that you have an origin access identity, you can create a distribution configured for private content. For more information, see Creating a Private Content Distribution (p. 77).

## Creating a Private Content Distribution

A distribution can serve either public or private content as specified by configuration values. To configure a distribution to serve private content, you use your AWS account, or a trusted AWS account you specify, to get a key pair. (If you already have an RSA key pair, you can upload the public key to AWS.) You then use the private key from the key pair to hash a policy statement; the result is a signature that you use to authenticate that the policy was generated by a trusted signer and has not been tampered with.

A private content distribution looks like a public content distribution, except that it has an `OriginAccessIdentity` element in the configuration. You must specify the value for the element using the following format: `origin-access-identity/cloudfront/ID`.

**To create a private content distribution**

• Create a new distribution that includes an `OriginAccessIdentity` element (or update an existing distribution to include the element).

The following example request creates a new private content distribution.

```
POST /2012-03-15/distribution HTTP/1.1
[Required headers]

<?xml version="1.0" encoding="UTF-8"?>
<DistributionConfig xmlns="http://cloudfront.amazonaws.com/doc/2012-03-15/">
  <S3Origin>
     <DNSName>myawsbucket.s3.amazonaws.com</DNSName>
    <OriginAccessIdentity>origin-access-identity/cloudfront/E127G7VEXAMPLE</Ori
ginAccessIdentity>
  </S3Origin>
  <CallerReference>20120229090000</CallerReference>
  <Comment>My comments</Comment>
```

```
  <Enabled>true</Enabled>
</DistributionConfig>
```

For information about updating an existing distribution, see Updating a Distribution's Configuration (p. 53).

The following request for a streaming distribution is similar to a distribution for static content.

```
POST /2012-03-15/streaming-distribution HTTP/1.1
[Required headers]

<?xml version="1.0" encoding="UTF-8"?>
<StreamingDistributionConfig xmlns="http://cloudfront.amazonaws.com/doc/2012-
03-15/">
  <S3Origin>
     <DNSName>myawsbucket.s3.amazonaws.com</DNSName>
    <OriginAccessIdentity>origin-access-identity/cloudfront/E127G7VEXAMPLE</Ori
ginAccessIdentity>
  </S3Origin>
  <CallerReference>20120229090000</CallerReference>
  <Comment>My comments</Comment>
  <Enabled>true</Enabled>
</StreamingDistributionConfig>
```

For information about updating an existing streaming distribution, see Updating a Distribution's Configuration (p. 53).

Now that you have created a distribution configured for private content, you need to set the ACLs on your Amazon S3 private content objects. For more information, see Updating Amazon S3 Bucket Policies or ACLs on Your Private Content Buckets or Objects (p. 78).

# Updating Amazon S3 Bucket Policies or ACLs on Your Private Content Buckets or Objects

After you create a private content distribution, you must update Amazon S3 bucket policies or ACLs to grant the CloudFront origin access identity the permissions necessary to access to the private content in Amazon S3. Note the following:

- You may find it easier to update Amazon S3 bucket policies than ACLs because you can add objects to the bucket without updating permissions. However, ACLs give you more fine-grained control because you're granting permissions on each object.

- If you updated a public-content distribution to serve private content, modify the bucket policy or any object ACLs as appropriate to ensure that the objects are not publicly available.

- Both for bucket policies and for ACLs, when you specify the CloudFront entity to which you are granting access, use the `S3CanonicalUserId` element that was returned when you created a CloudFront origin access identity.

## Updating Amazon S3 Bucket Policies

Using either the AWS Management Console or the Amazon S3 API, change the Amazon S3 bucket policy to allow the CloudFront origin access identity to access objects in the bucket. For more information, go to Using Bucket Policies in the *Amazon Simple Storage Service Developer Guide*. For an example, see "Granting Permission, Using Canonical ID, to a CloudFront Origin Identify" in the topic Example Cases for Amazon S3 Bucket Policies, also in the *Amazon Simple Storage Service Developer Guide*.

## Updating Amazon S3 ACLs

Using either the AWS Management Console or the Amazon S3 API, change the Amazon S3 ACL to give CloudFront `READ` permission on each object that the CloudFront distribution serves. For more information, go to Using ACLs in the *Amazon Simple Storage Service Developer Guide*.

You can also change the ACLs using code and one of the AWS SDKs. For an example, see the downloadable sample code in Create a URL Signature Using C# and the .NET Framework (p. 101).

## What's Next?

After you grant the CloudFront origin access identity the permissions necessary to access your Amazon S3 content, you may want to restrict end-user access to your distribution and create a signed URL. For more information, go to Restricting End User Access (p. 80).

# Restricting End User Access

**Topics**

Restricting end user access involves creating a key pair, modifying your distribution to require signed URLs, and then creating the signed URL. This section describes these processes.

## Creating a Key Pair

*Signing a URL* is the process of creating an RSA digital signature using an RSA key and a policy statement. This section describes how to get the *key pair* consisting of a private key and a public key. AWS keeps the public key, and you keep the private key and use it to sign the URLs.

> ⚠️ **Important**
>
> The key pair is *not* an X.509 certificate and private key. It's an RSA key pair.
>
> If you're an Amazon EC2 user, you probably already have at least one RSA key pair, which you use to connect to your EC2 instances through SSH or Windows Remote Desktop, but you can't reuse your EC2 key pairs with CloudFront because the key pair ID is not supplied. If you want to use your own key pair, see the procedure that follows for uploading your own public key to the AWS website.

### Using Your Own Key Pair

If you have a key pair that you want to use, you can upload the public key to AWS (you keep the private key). The public key must be an RSA key encoded in PEM format.

**To upload your own public key**

1. From the Amazon Web Services website at http://aws.amazon.com, point to **Your Account** and click **Security Credentials**.

2. Log in to your AWS account.
   The **Security Credentials** page is displayed.

3. In the **Access Credentials** section of the page, click the **Key Pairs** tab.

4. In the **Amazon CloudFront Key Pairs** area, click **Upload Your Own Key Pair**.

5. Follow the instructions presented to upload your public key.

### Using a Key Pair Generated by AWS

If you don't already have a key pair, you can have AWS generate a pair and automatically associate the public key with your AWS account.

**To have AWS create a key pair for you**

1. From the Amazon Web Services website at http://aws.amazon.com, point to **Your Account** and click **Security Credentials**.

2. Log in to your AWS account.
   The **Security Credentials** page is displayed.

3. In the **Access Credentials** section of the page, click the **Key Pairs** tab.

4. In the **Amazon CloudFront Key Pairs** area, click **Create a New Key Pair**.

   Your new public and private key are generated, along with an ID for the key pair. Amazon keeps the public key and gives you the private key.

5. From the dialog box, download your private key file to a local directory, and record the corresponding key pair ID.

You should keep your private key file secure. Make sure to set the permissions on the file so only you can read it. For a Linux/UNIX system, use `chmod 600`. To set the permission on a Windows system, right-click the file and set the file's security properties appropriately.

The next step is to configure your distribution to require signed URLs. For more information, see Requiring Signed URLs (p. 81).

# Requiring Signed URLs

You must configure your private content distribution to specify that URLs must be signed, and include the accounts that can sign them. Up to five AWS accounts other than your own can sign URLs for a single distribution. Each AWS account that you authorize must create and use its own key pair. For more information, see Creating a Key Pair (p. 80). A signed URL includes the signing key ID in the URL so that AWS can identify the signer account.

### To specify that URLs must be signed

- Add a `TrustedSigners` element to the configuration for the download distribution or streaming distribution.

### To specify who can sign URLs

1. If you want to sign URLs yourself, add an empty `Self` child element to the `TrustedSigners` element. We don't assume that you do, so you must explicitly give yourself permission.

2. Add an `AwsAccountNumber` child element to the `TrustedSigners` element for each AWS account, other than your own, to which you want to give signing authority (there is a limit of five). Note that you must remove the dashes from the account numbers.

The AWS account number is displayed in the top right corner of the account owner's **Account Activity** page at http://aws.amazon.com.

The following example request creates a private content distribution, and authorizes you and two other AWS accounts to create signed URLs for the distribution.

```
POST /2012-03-15/distribution HTTP/1.1
[Required headers]

<?xml version="1.0" encoding="UTF-8"?>
<DistributionConfig xmlns="http://cloudfront.amazonaws.com/doc/2012-03-15/">
  <S3Origin>
     <DNSName>myawsbucket.s3.amazonaws.com</DNSName>
    <OriginAccessIdentity>origin-access-identity/cloudfront/E127G7VEXAMPLE</Ori
ginAccessIdentity>
  </S3Origin>
  <CallerReference>20120229090000</CallerReference>
  <Comment>My comments</Comment>
```

```
  <Enabled>true</Enabled>
  <TrustedSigners>
    <Self/>
    <AwsAccountNumber>111122223333</AwsAccountNumber>
    <AwsAccountNumber>444455556666</AwsAccountNumber>
  </TrustedSigners>
  </DistributionConfig>
```

If you are working with a streaming distribution, the request for a private streaming distribution is similar.

```
POST /2012-03-15/streaming-distribution HTTP/1.1
[Required headers]

<?xml version="1.0" encoding="UTF-8"?>
<StreamingDistributionConfig xmlns="http://cloudfront.amazonaws.com/doc/2012-
03-15/">
   <S3Origin>
     <DNSName>mystreamingbucket.s3.amazonaws.com</DNSName>
     <OriginAccessIdentity>origin-access-identity/cloudfront/E127G7VEXAMPLE</Ori
ginAccessIdentity>
   </S3Origin>
  <CallerReference>20120229090000</CallerReference>
  <Comment>My comments</Comment>
  <Enabled>true</Enabled>
  <TrustedSigners>
    <Self/>
    <AwsAccountNumber>111122223333</AwsAccountNumber>
    <AwsAccountNumber>444455556666</AwsAccountNumber>
  </TrustedSigners>
</StreamingDistributionConfig>
```

Once you've specified trusted signers, you should verify that the signers are *active*. For a trusted signer to be active, both of the following must be true:

- The AWS account must have at least one *active* key pair (you can set a key pair to *inactive* when you rotate your keys; for more information, go to Access Credential Rotation).
- CloudFront must be aware of the active key pair (after you create a key pair, there can be a short period of time before CloudFront is aware the key pair exists).

To determine the *active trusted signers* for a distribution, get the distribution's information, not just the configuration, but the entire distribution. The response includes an ActiveTrustedSigners element that lists the ID of each signer and the active key pairs associated with the trusted signer's AWS account. If a signer doesn't have an active key pair, CloudFront will not recognize that account as a signer.

The following example response shows that you yourself have two active key pairs, and the AWS account with ID 111122223333 has one active key pair. However, the third trusted signer (with account ID 444455556666) doesn't yet have an active key pair (no KeyPairId appears for that signer) and so the account can't create working signatures.

```
200 OK
ETag: E2QWRUHEXAMPLE
x-amz-request-id: request_id

<Distribution xmlns="http://cloudfront.amazonaws.com/doc/2012-03-15/">
   <Id>EDFDVBD6EXAMPLE</Id>
```

```
      <Status>Deployed</Status>
      <LastModifiedTime>2009-11-19T19:37:58Z</LastModifiedTime>
      <DomainName>d111111abcdef8.cloudfront.net</DomainName>
      <ActiveTrustedSigners>
          <Signer>
              <Self/>
              <KeyPairId>APKAI72T5DYBXEXAMPLE</KeyPairId>
          </Signer>
          <Signer>
              <AwsAccountNumber>111122223333</AwsAccountNumber>
              <KeyPairId>APKA9ONS7QCOWEXAMPLE</KeyPairId>
          </Signer>
          <Signer>
              <AwsAccountNumber>444455556666</AwsAccountNumber>
          </Signer>
      </ActiveTrustedSigners>
      <DistributionConfig>
          <S3Origin>
              <DNSName>myawsbucket.s3.amazonaws.com</DNSName>
              <OriginAccessIdentity>E74FTE3AEXAMPLE</OriginAccessIdentity>
          </S3Origin>
          <CallerReference>20120229090000</CallerReference>
          <Comment>My comments</Comment>
          <Enabled>true</Enabled>
          <TrustedSigners>
              <Self/>
              <AwsAccountNumber>111122223333</AwsAccountNumber>
              <AwsAccountNumber>444455556666</AwsAccountNumber>
          </TrustedSigners>
      </DistributionConfig>
</Distribution>
```

The response for a streaming distribution would be identical except that in place of the elements `Distribution` and `DistributionConfig`, the response would contain `StreamingDistribution` and `StreamingDistributionConfig`, respectively.

Next, you need to create your signed URL. For more information, see Creating a Signed URL (p. 83).

# Creating a Signed URL

**Topics**

- Overview of Signed URLs (p. 83)
- Creating a Policy Statement and a Signature (p. 85)
- Signed URL Examples (p. 91)

This section gives an overview of signed URLs, describes how to create a policy statement and a signed URL, and provides some examples of how to create a signed URL.

## Overview of Signed URLs

A signed URL is composed of several parts. Following is an example of a CloudFront signed URL that uses a custom policy.

| | |
|---|---|
| ❶ | The CloudFront domain name with the file or media stream identifier. This is your base URL. |
| ❷ | The policy statement request parameter. The policy statement was Base64-encoded, and several characters that are invalid in URL request parameters were replaced with valid characters. For more information, see Creating a Policy Statement and a Signature (p. 85). |
| ❸ | The signature request parameter. The signature was Base64-encoded, and several characters that are invalid in URL request parameters were replaced with valid characters. For more information, see Creating a Policy Statement and a Signature (p. 85). |
| ❹ | The Key-Pair-Id request parameter. This is the ID for the key pair that is associated with the account you are using to create the signature. |

A canned policy, the simplest form of the signed URL, uses the signature, the ID, and an expiration date, but omits the encoded policy statement, because the only restriction is the expiration date. A custom policy requires the signature and policy request parameters, but not the expiration date, which, in the example above, is contained in the policy statement.

⚠️ **Important**

Your signed URL must not contain any whitespace. You might have to include escape characters in the string in application code.

The segments of a signed URL are described in the following examples.

The first segment is the CloudFront distribution domain name and the file to be retrieved, which in this case is demo.txt. The question mark (?) indicates that request parameters will follow.

```
http://dekrsqm7un9m0.cloudfront.net/demo.txt?
```

The following Base64 encoded string is the policy statement as a request parameter. Characters that are not valid in a request parameter have been replaced with valid characters. For more information, see Creating a Policy Statement and a Signature (p. 85).

```
Policy=eyANCiAgICJTdGF0ZW1lbnQiOiBbeyANCiAgICAgICJSZXNvdXJjZSI6Imh0dHA6Ly9kemJlJlc
3FtN3VuMW0wLmNsb3VkZnJvbnQubmV0L2RlbW8ucGhwIiwgDQogICAgICAiQ29uZGl0aW9uIjp7IA0KI
CAgICAgICAgIklwQWRkcmVzcyI6eyJBV1M6U291cmNlSXAiOiIyMDcuMTcxLjE4MC4xMDEvMzIifSwNC
iAgICAgICAgICJEYXRlR3JlYXRlclRoYW4iOnsiQVdTOkVwb2NoVGltZSI6MTI5Njg2MDE3Nn0sDQogI
CAgICAiRGF0ZUxlc3NUaGFuIjp7IkFXUzpFcG9jaFRpbWUiOjEyOTY4NjAyMjZ9DQogICAgICB9IA0KIC
A0KICAgfV0gDQp9DQo=
```

The next request parameter, indicated by the ampersand (`&`), is the Base64-encoded signature. As with the policy statement, characters that are not valid in a request parameter have been replaced with valid characters.

```
&Signature=nitfHRCrtziwO2HwPfWw~yYDhUF5EwRunQA-j19DzZrvDh6hQ73lDx~-ar3UocvvRQVw6E
kC~GdpGQyyOSKQim-TxAnW7d8F5Kkai9HVx0FIu-5jcQb0UEmatHw3FTxb3ReXySpLSMj0yCd3ZAB4Uc
BCAqEijkytL6f3EXAMPLE=
```

The `Key-Pair-Id` request parameter is always required.

```
&Key-Pair-Id=APLDH2VGALRTSEXAMPLE
```

# Creating a Policy Statement and a Signature

**Topics**

A policy statement specifies the restrictions on a signed URL. There are two types of policy statements: *canned* and *custom*. A canned policy statement is short and specifies only one condition: an end date after which the URL is invalid. A custom policy statement can include the start date, the end date, and the IP address or range of IP addresses for which the URL is valid. For both types of policy statement, the policy is defined in Java Script Object Notation (JSON) in UTF-8 format.

A signature is created by SHA1-hashing the policy statement and then encrypting the result by using RSA and the private key for your AWS account or for a trusted AWS account that you specify.

> **Note**
>
> Sample signature code is available at AWS Developer Resources Sample Code & Libraries, and in the section Signature Code, Examples, and Tools (p. 97). Additionally, CloudFront provides a Perl script you can use to create an URL Signature. For more information about the Perl script, see Create a URL Signature Using Perl (p. 97).

**To create a signed URL**

1. Create a policy statement. For more information, see the applicable section, Canned Policy (p. 86) or Custom Policy (p. 88).

2. If you are using a canned policy in the signed URL, skip to the next step.

   If you are using a custom policy, Base64-encode the policy statement, and replace invalid characters with valid characters to make the string URL-safe, as indicated in the following table. For an example, see Signed URL Examples (p. 91).

   | Invalid characters | Valid characters |
   |---|---|
   | **+** | **-** |
   | **=** | _ |
   | **/** | **~** |

3.  Create a digital signature by SHA1-hashing the policy statement and RSA-encrypting the result using the private key for your AWS account or for a trusted AWS account that you specify. Then Base64-encode the result, and replace invalid characters with valid characters to make the string URL-safe, as indicated in the table in the previous step. For an example, see Signed URL Examples (p. 91).

4.  Concatenate the CloudFront URL and the applicable parameters, depending on whether you are using a canned policy or a custom policy.

> ⚠️ **Important**
>
> Your signed URL will not work if it cannot access the object origin. Ensure that you have granted read access to the private content to your CloudFront origin access identity. You do this by modifying the Amazon S3 ACL on each of the objects, not on the bucket. For more information, see Updating Amazon S3 Bucket Policies or ACLs on Your Private Content Buckets or Objects (p. 78).

The main part of this section covers how to correctly create the string that you sign. When the end user clicks the URL, the signature is verified and CloudFront evaluates the contents of the policy to determine if the end user is authorized access to the object specified by the URL. The following table describes the two types of policies you can use: *canned* or *custom*.

| Type of Policy | Description |
| --- | --- |
| Canned | Lets you restrict access to a single URL based only on expiration time. You don't include the policy in the URL itself, so a canned policy results in a shorter URL. Because you don't include the policy in the URL, CloudFront constructs a *canned* policy based on information in the URL itself, and uses the canned policy to both validate the signature and determine if the end user has access to the content. |
| | For more information about using a canned policy, see Canned Policy (p. 86). |
| Custom | Lets you restrict access to one or more objects based on: end user IP address, a start time for access, and an expiration time for access. You must include the policy in the URL itself, so a custom policy results in a longer URL than a canned policy. CloudFront uses the policy statement to validate the signature and determine if the user has access to the content. |
| | For more information, see Custom Policy (p. 88). |

## Canned Policy

A signed URL that uses a canned policy consists of a regular CloudFront URL, plus the three request parameters listed in the following table. You can use canned policies with HTTP and RTMP distributions.

| Parameter | Description | Required |
| --- | --- | --- |
| *Expires* | The expiration time of the URL, in epoch or UNIX time (number of seconds since January 1, 1970; e.g., 1258237200). Type: String | Yes |
| *Signature* | A URL-safe version of the signature. Type: String | Yes |
| *Key-Pair-Id* | The key ID of the signing private key. Type: String | Yes |

The *Signature* value is an RSA-SHA1 digital signature of the following JSON policy, with the RESOURCE and EXPIRES values replaced as described in the table that follows the example.

**Example Canned Policy**

```
{"Statement":[{"Resource":"RESOURCE","Condition":{"DateLessThan":{"AWS:Epoch
Time":EXPIRES}}}]}
```

> ⚠️ **Important**
>
> For the signature you include in the URL to match the signature CloudFront constructs based on the canned policy, you must make sure the policy you construct looks exactly like the preceding policy. That is, you must use your own valid values for RESOURCE and EXPIRES, and you must remove any whitespace. You might have to include escape characters in the string in application code.

The following table describes the values to substitute in the policy.

| Value in Policy | Substitute with... |
| --- | --- |
| RESOURCE (HTTP) | The URL without the CloudFront request parameters *Expires*, *Signature*, and *Key-Pair-Id* parameters that will be created in the following steps. Retain any query parameters you've included in the URL. For example, if the full URL created by the signing process is:<br><br>`http://d111111abcdef8.cloudfront.net/download/hori`<br>`zon.jpg?large=yes&license=yes &Expires=1258237200&Signature=TBD`<br>`&Key-Pair-Id=PK12345EXAMPLE`<br><br>The value for RESOURCE is:<br><br>`http://d111111abcdef8.cloudfront.net/download/hori`<br>`zon.jpg?large=yes&license=yes`<br><br>If there are no request parameters, don't include the question mark in the value. Put the value inside quotation marks in the policy. |

| Value in Policy | Substitute with... |
|---|---|
| RESOURCE (RTMP) | With HTTP, a full URL uniquely describes an object. You can include the URL in the signature. The content of a streaming distribution, however, cannot always be described by a valid URL. In a streaming distribution, you only use the stream name to create a signature. For example, if your stream including the signature is:<br><br>`example/mp3_name.mp3?Expires=1258237200&Signature=TBD&Key-Pair-Id=PK12345EXAMPLE`<br><br>The value for `RESOURCE` is:<br><br>`example/mp3_name`<br><br>For streaming distributions, you do not include a prefix, such as mp3: or mp4:, for the resource name in the policy.<br>Also, when referencing an MPEG file, you might have to omit the file extension for the URL enclosed in the signature. For example, you use `mp3_name` instead of `mp3_name.mp3`. |
| EXPIRES | The value of the *Expires* request parameter. Because the value is a number, which is specified in epoch seconds or UNIX time, you don't have to put quotation marks around it in the policy. For HTTP, we check the Expires value at the beginning of the HTTP request. For streaming content, CloudFront enforces the expiration only for play events. It is possible, for example, to play streaming content once, but when trying to replay it, the play fails because the UNIX time exceeded the value for *Expires*. |

To create the signature, you SHA1-hash the policy statement, RSA-encrypt the result using the private key for your AWS account or for a trusted AWS account that you specify, remove whitespace, Base64-encode that result, and replace characters that are invalid in a URL request parameter (+, =, /) with characters that are valid (-, _, and ~, respectively). For an example, see Signed URL Examples (p. 91).

How you implement encryption of the signature depends on programming language and platform. This documentation provides examples in Perl, PHP, and C#. There are links to development resources for Java. After you read the following sections that explain the details of policy statements for custom URL restrictions, look at the signature code examples in Signature Code, Examples, and Tools (p. 97).

⚠ **Important**

The CloudFront process for creating a signature uses SHA1 and RSA. S3 and other AWS services use HMAC-SHA1.

Base64 encode the signature, and replace +, =, and / with -, _, and ~, respectively, to make it URL safe before including it in the URL. (For more information, see Signature Code, Examples, and Tools (p. 97).)

For an example of a complete URL that uses a canned policy, see Signed URL Examples (p. 91).

## Custom Policy

A signed URL that uses a custom policy consists of a regular CloudFront URL, plus the three request parameters listed in the following table. Notice that no *Expires* parameter is used, as a canned policy would. A *Policy* parameter is required instead.

| Parameter | Description | Required |
|---|---|---|
| *Policy* | A URL-safe version of the policy. The presence of the *Policy* parameter in the URL indicates a custom policy instead of a canned policy.<br>Type: String | Yes |
| *Signature* | A URL-safe version of the signature<br>Type: String | Yes |
| *Key-Pair-Id* | The key ID of the signing private key.<br>Type: String | Yes |

The policy you use to create the signature is a JSON document in UTF-8 format that specifies the resource and any conditions for accessing the resource. Use the policy format shown in the following example and the parameters listed in the following table.

The following example is a policy statement that allows access to the game_download.zip object if the end user's IP address is within the 192.0.2.0/24 IP address range, and if the end user's request for the object comes in before 11/14/2011 at 10:20 p.m., as specified in epoch time in seconds. There should be one newline character at the end of the custom policy statement.

**Example Custom Policy**

```
{
   "Statement": [{
      "Resource":"http://d111111abcdef8.cloudfront.net/game_download.zip",
      "Condition":{
         "IpAddress":{"AWS:SourceIp":"192.0.2.0/24"},
         "DateLessThan":{"AWS:EpochTime":1258237200}
      }
   }]
}
```

⚠️ **Important**

If you're familiar with Amazon S3 browser-based POSTs, the policy format you use there differs from the policy format you use here for CloudFront private objects.

If you're familiar with Amazon SQS access control, CloudFront uses the same format for its policies, but limits how you can use the syntax only for the following specific cases.

The following table describes the parameters you can specify in custom policy.

| Parameter | Description | Required |
|---|---|---|
| `Resource` (HTTP) | The URL to the cached object itself. This is the *CloudFront URL* (using the CloudFront domain name), not the URL to the object in the Amazon S3 bucket.<br><br>The value must include the `http://`, and must match the resource specified in the URL. You can use multi-character match wild cards (*) or a single-character match wild card (?) anywhere in the string. For example, the value could be `http://d111111abcdef8.cloudfront.net/*game_download.zip*`. This would include (for example) `example_game_download.zip?license=yes`.<br><br>Omitting this parameter gives the end user access to *all* the objects belonging to any distribution associated with the key pair used to sign the URL. | Optional |
| `Resource` (RTMP) | When you use streaming content as your resource, use the stream name in the policy, for example, `example/mp3_name.mp3?Expires=1258237200 &Signature=TBD&Key-Pair-Id=PK12345EXAMPLE`.<br><br>When referencing an MPEG file, you might need to omit the file extension for the URL enclosed in the signature. For example, you use `mp3_name` instead of `mp3_name.mp3`. | Optional |
| `DateLessThan` | This is the only required parameter. It specifies an expiration date and time for the URL, using the format `"AWS:EpochTime":`*value* in seconds, and with no quotation marks. We require this value to prevent permanent access to any private content. | Required |
| `DateGreaterThan` | Specifies an optional start date and time for the URL, using the format `"AWS:EpochTime":`*value* in seconds, and with no quotation marks. | Optional |
| `IpAddress` | Specifies the IP address of the client making the GET request, using the format `"AWS:SourceIp":"`*value*`"`.<br><br>It must be in standard CIDR format (for example, 10.52.176.0/24). For more information, go to RFC 4632. You can specify only a single value for the condition. For example, you can't set the policy to allow access if the client's IP address is in one of two different ranges.<br><br>To allow access to all IP addresses, omit this parameter. | Optional |

The parameter names must be specified in the policy exactly as shown in the preceding table (no abbreviations like `datelt` for `DateLessThan` are accepted). The order of the parameters in the policy doesn't matter. Specify the conditions (`DateLessthan`, `DateGreaterThan`, and `IpAddress`) as part of the `Condition` section in the policy as shown in the examples in Signed URL Examples (p. 91).

The square brackets that enclose the statement's contents, as shown in the preceding example policy, are required for the policy to be valid.

You calculate the *Signature* request parameter using an RSA-SHA1 digital signature of the policy statement (the policy must be in UTF-8 format before signing).

> 📝 **Note**
>
> Signature encryption depends on platform and code language options. For more information about signature generation in various code languages, see Signature Code, Examples, and Tools (p. 97).

Base64 encode the signature, and replace +, =, and / with -, _, and ~, respectively, to make it URL safe. For more information, see Signature Code, Examples, and Tools (p. 97).

## Signed URL Examples

This section shows example signatures based on the non-working credentials in the following table. You can download the example credentials zipped in the CloudFront_PrivateContent_SignatureExamples.zip file.

| Credential | Value |
|---|---|
| Public Key | <pre>-----BEGIN PUBLIC KEY-----<br>MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDA7ki9gI/lRygIoOjV1yymgx6F<br>YFlzJ+z1ATMaLo57nL57AavWhb68HYY8EA0GJU9xQdMVaHBogF3eiCWYXSUZCWM/<br>+M5+ZcdQraRRScucmn6g4EvY2K4W2pxbqH8vmUikPxir41EeBPLjMOzKvbzzQy9e<br>/zzIQVREKSp/7y1myEXAMPLE<br>-----END PUBLIC KEY-----</pre> |
| Private Key | <pre>-----BEGIN RSA PRIVATE KEY-----<br>MIICXQIBAAKBgQDA7ki9gI/lRygIoOjV1yymgx6FYFlzJ+z1ATMaLo57nL57AavW<br>hb68HYY8EA0GJU9xQdMVaHBogF3eiCWYXSUZCWM/+M5+ZcdQraRRScucmn6g4EvY<br>2K4W2pxbqH8vmUikPxir41EeBPLjMOzKvbzzQy9e/zzIQVREKSp/7y1mywIDAQAB<br>AoGABc7mp7XYHynuPZxChjWNJZIq+A73gm0ASDv6At7F8Vi9r0xUlQe/v0AQS3yc<br>N8QlyR4XMbzMLYk3yjxFDXo4ZKQtOGzLGteCU2srANiLv26/imXA8FVidZftTAtL<br>viWQZBVPTeYIA69ATUYPEq0a5u5wjGyUOij9OWyuy01mbPkCQQDluYoNpPOekQ0Z<br>WrPgJ5rxc8f6zG37ZVoDBiexqtVShIF5W3xYuWhW5kYb0hliYfkq15cS7t9m95h3<br>1QJf/xI/AkEA1v9l/WN1a1N3rOK4VGoCokx7kR2SyTMSbZgF9IWJNOugR/WZw7HT<br>njipO3c9dy1Ms9pUKwUF46d7049ck8HwdQJARgrSKuLWXMyBH+/l1Dx/I4tXuAJI<br>rlPyo+VmiOc7b5NzHptkSHEPfR9slOK0VqjknclqCJ3Ig86OMEtEFBzjZQJBAKYz<br>470hcPkaGk7tKYAgP48FvxRsnzeooptURW5E+M+PQ2W9iDPPOX9739+Xi02hGEWF<br>B0IGbQoTRFdE4VVcPK0CQQCeS84lODlC0Y2BZv2JxW3Osv/WkUQ4dslfAQl1T303<br>7uwwr7XTroMv8dIFQIPreoPhRKmd/SbJzbiKfEXAMPLE<br>-----END RSA PRIVATE KEY-----</pre> |
| Key-Pair-Id | PK12345EXAMPLE |

Using the OpenSSL package, you can calculate the request parameters as shown in the following examples. For information about OpenSSL, go to http://www.openssl.org.

The following command creates the URL-safe *Policy* value.

```
% cat policy | openssl base64 | tr '+=/' '-_~'
```

The following command creates the URL-safe *Signature* value.

```
% cat policy | openssl sha1 -sign private-key.pem | openssl base64 | tr '+=/'
'-_~'
```

> 📋 **Note**
>
> You must remove whitespace from the resulting Base64 encoding.

For code examples that demonstrate creating a signature in several programming languages see Signature Code, Examples, and Tools (p. 97)

### Example Canned Policy

The following canned policy example gives any user with the signed URL access to `http://d604721fxaaqy9.cloudfront.net/horizon.jpg` before Mon, 14 Nov 2011 22:20:00 GMT.

The original URL is the CloudFront URL and request parameters.

```
http://d604721fxaaqy9.cloudfront.net/horizon.jpg?large=yes&license=yes
```

The policy statement that will be hashed and encrypted into the signature uses the original URL and an expiration time in epoch/UNIX seconds. If you copy and paste this example, remove any whitespace, and replace the URL and expiration time with your own values.

```
{"Statement":[{"Resource":"http://d604721fxaaqy9.cloudfront.net/horizon.jpg
?large=yes&license=yes","Condition":{"DateLessThan":{"AWS:Epoch
Time":1258237200}}}]}
```

The signature is the result of SHA1 hashing, RSA encryption, and Base64 encoding of the result. You must also replace +, =, and / with -, _, and ~, respectively, to make the value URL safe. For more information about this process, see Signature Code, Examples, and Tools (p. 97).

```
Signature = Nql641NHEUkUaXQHZINK1FZ~SYeUSoBJMxjdgqrzIdzV2gyEXPDNv0pYdWJkflDKJ3xIu
7lbwRpSkG98NBlgPi4ZJpRRnVX4kXAJK6tdNx6FucDB7OVqzcxkxHsGFd8VCG1BkC-Afh9~lOCMIY
HIaiOB6~5jt9w2EOwiEXAMPLE_
```

Following is the full URL for the authorized user.

```
http://d604721fxaaqy9.cloudfront.net/horizon.jpg?large=yes&license=yes&Ex
pires=1258237200&Signature=Nql641NHEUkUaXQHZINK1FZ~SYeUSoBJMxjdgqrzIdzV2gyEX
PDNv0pYdWJkflDKJ3xIu7lbwRpSkG98NBlgPi4ZJpRRnVX4kXAJK6tdNx6FucDB7OVqzcxkxHsG
Fd8VCG1BkC-Afh9~lOCMIYHIaiOB6~5jt9w2EOwiEXAMPLE_&Key-Pair-Id=PK12345EXAMPLE
```

### Example Custom Policy 1

The objective of the following custom policy example is to grant the network 145.168.143.0/24 access to all the objects in the `training` directory before Mon, 14 Nov 2011 22:20:00 GMT.

The original URL, as follows, specifies a particular object (training/orientation.avi), but the policy creates a signed URL that can access multiple objects.

```
http://d604721fxaaqy9.cloudfront.net/training/orientation.avi
```

The policy statement to sign includes an asterisk wildcard character to indicate any item in the bucket. To get the proper syntax for the examples, download the following policy (zipped in the CloudFront_PrivateContent_SignatureExamples.zip file). Note that the string used to calculate the signature includes whitespace. There should be one newline character after the final } bracket.

```
{
    "Statement": [{
        "Resource":"http://d604721fxaaqy9.cloudfront.net/training/*",
        "Condition":{
            "IpAddress":{"AWS:SourceIp":"145.168.143.0/24"},
            "DateLessThan":{"AWS:EpochTime":1258237200}
        }
    }]
}
```

The URL-safe policy is Base64 encoded into the result following `Policy=`.

```
Policy=eyAKICAgIlN0YXRlbWVudCI6IFt7IAogICAgICAiUmVzb3VyY2UiOiJodHRwOi8vZDYwND
cyMWZ4YWFxeTkuY2xvdWRmcm9udC5uZXQvdHJhaW5pbmcvKiIsIAogICAgICAiQ29uZ
Gl0aW9uIjp7IAogICAgICAgICAiSXBBZGRyZXNzIjp7IkFXUzpTb3VyY2VJcCI6IjE0NS4xNjguMT
QzLjAvMjQifSwgCiAgICAgICAgICJEYXRlTGVzc1RoYW4iOnsiQVdTOkVwb2NoVGltZSI6MTI1
ODIzNzIwMH0gICAgICAKICAgICAgfSAKICAgfSBdEXAMPLE
```

The signature is the result of SHA1 hashing, RSA encryption, and Base64 encoding of the result. You must also replace +, =, and / with -, _, and ~, respectively, to make the value URL safe. For more information about this process, see .

```
Signature=cPFtRKvUfYNYmxek6ZNs6vgKEZP6G3Cb4cyVt~FjqbHOnMdxdT7eT6pYmhHYzuDsFH4Jps
ctke2Ux6PCXcKxUcTIm8SO4b29~1QvhMl-CIojki3Hd3~Unxjw7Cpo1qRjtvrimW0DPZBZYHFZtiZX
saPt87yBP9GWnTQoEXAMPLE_
```

Following is the full URL for the authorized user.

```
http://d604721fxaaqy9.cloudfront.net/training/orientation.avi?Policy=eyAKICAg
IlN0YXRlbWVudCI6IFt7IAogICAgICAiUmVzb3VyY2UiOiJodHRwOi8vZDYwNDcyMWZ4Y
WFxeTkuY2xvdWRmcm9udC5uZXQvdHJhaW5pbmcvKiIsIAogICAgICAiQ29uZGl0aW9uIjp7IAog
ICAgICAgICAiSXBBZGRyZXNzIjp7IkFXUzpTb3VyY2VJcCI6IjE0NS4xNjguMTQzLjAvMjQifSwgCiAgI
CAgICAgICJEYXRlTGVzc1RoYW4iOnsiQVdTOkVwb2NoVGltZSI6MTI1ODIzNzIwMH0gICAgICAKICAgI
CAgfSAKICAgfSBdEXAMPLE&Signature=cPFtRKvUfYNYmxek6ZNs6vgKEZP6G3Cb4cyVt~FjqbHOnM
dxdT7eT6pYmhHYzuDsFH4Jpsctke2Ux6PCXcKxUcTIm8SO4b29~1QvhMl-CIojki3Hd3~Unxjw7
Cpo1qRjtvrimW0DPZBZYHFZtiZXsaPt87yBP9GWnTQoEXAMPLE_&Key-Pair-Id=PK12345EXAMPLE
```

For code examples that demonstrate creating a signature in several programming languages see Signature Code, Examples, and Tools (p. 97)

### Example Custom Policy 2

The objective of the following custom policy is to grant the IP addresses 216.98.35.1/32 access to all the objects belonging to any distribution that the specified key pair ID is associated with. The objects are to be available only between Sat, 30 Apr 2011 06:43:10 GMT and Sun, 16 Oct 2011 06:31:56 GMT.

The original URL specifies a particular object, downloads/pictures.tgz, but the policy covers potentially many.

```
http://d84l721fxaaqy9.cloudfront.net/downloads/pictures.tgz
```

The policy statement to sign includes an asterisk wildcard character to indicate any item referenced by any distribution associated with the key pair used to sign the URL. To get the proper syntax for the examples, download the following policy (zipped in the CloudFront_PrivateContent_SignatureExamples.zip file). Note that the string used to calculate the signature includes whitespace. There should be one newline character after the final } bracket.

```
{
    "Statement": [{
        "Resource":"http://*",
        "Condition":{
            "IpAddress":{"AWS:SourceIp":"216.98.35.1/32"},
            "DateGreaterThan":{"AWS:EpochTime":1241073790},
            "DateLessThan":{"AWS:EpochTime":1255674716}
        }
    }]
}
```

The URL-safe policy is Base64 encoded into the result following `Policy=`.

```
Policy=eyAKICAgIlN0YXRlbWVudCI6IFt7IAogICAgICAiUmVzb3VyY2UiOiJodHRwOi8
vKiIsIAogICAgICAiQ29uZGl0aW9uIjp7IAogICAgICAgICAiSXBBZGRyZXNzIjp7IkFFXU
zpTb3VyY2VJcCI6IjIxNi45OC4zNS4xLzMyIn0sCiAgICAgICAgICJEYXRlR3JlYXRlclRoYW4iOnsiQV
dTOkVwb2NoVGltZSI6MTI0MTA3Mzc5MH0sCiAgICAgICAgICJEYXRlTGVzc1RoYW4iOnsiQVdTOkVwb2N2w
b2NoVGltZSI6MTI1NTY3NDcxNn0KICAgICAgfSAKICAgIGfSAKICAgIGfEXAMPLE
```

The signature is the result of SHA1 hashing, RSA encryption, and Base64 encoding of the result. You must also replace +, =, and / with -, _, and ~, respectively, to make the value URL safe. For more information about this process, see Signature Code, Examples, and Tools (p. 97).

```
Signature=rc~5Qbbm8EJXjUTQ6Cn0LAxR72g1DOPrTmdtfbWVVgQNw0q~KHUAmB
a2Zv1Wjj8dDET4XSL~Myh44CLQdu4dOH~N9huH7QfPSR~O4tIOS1WWcP~2JmtVPoQyLlEc8YHRCuN3nVN
ZJ0m4EZcXXNAS-0x6Zco2SYx~hywTEXAMPLE_
```

Following is the full URL for the authorized user.

```
http://d84l721fxaaqy9.cloudfront.net/downloads/pictures.tgz?Policy=eyAKICA
gIlN0YXRlbWVudCI6IFt7IAogICAgICAiUmVzb3VyY2UiOiJodHRwOi8vKiIsIAogICAgICAiQ
29uZGl0aW9uIjp7IAogICAgICAgICAiSXBBZGRyZXNzIjp7IkFFXUzpTb3VyY2VJcCI6IjIxNi45OC
4zNS4xLzMyIn0sCiAgICAgICAgICJEYXRlR3JlYXRlclRoYW4iOnsiQVdTOkVwb2NoVGltZSI6MTI
0MTA3Mzc5MH0sCiAgICAgICAgICJEYXRlTGVzc1RoYW4iOnsiQVdTOkVwb2NoVGltZSI6MTI1NTY3NND
cxNn0KICAgICAgfSAKICAgIGfSAKICAgIGfEXAMPLE&Signature=rc~5Qbbm8EJXjUTQ6Cn0LAxR72g1DOPrTmdtf
```

```
bWVVgQNw0q~KHUAmBa2Zv1Wjj8dDET4XSL~Myh44CLQdu4dOH~N9huH7QfPSR~O4tIOS1WWcP~2JmtV
PoQyLlEc8YHRCuN3nVNZJ0m4EZcXXNAS-0x6Zco2SYx~hywTEXAMPLE_&Key-Pair-
Id=PK12345EXAMPLE
```

For code examples that demonstrate creating a signature in several programming languages see Signature Code, Examples, and Tools (p. 97).

# Signature Code, Examples, and Tools

This section includes downloadable application samples that demonstrate how to create signature for CloudFront private distributions. Examples are available in Perl, PHP, C#, as well as links to OpenSSL and Java implementations. All the samples can be used to create signed URLs. The Perl script runs on Linux/Mac platforms. The PHP example will work on any server that runs PHP. The C# example uses the .NET Framework.

**Topics**

## Create a URL Signature Using Perl

The Perl script creates the signature for private content from command line arguments that specify the CloudFront URL, the path to the private key of the signer, the key ID, and an expiration date for the URL. The tool can also decode signed URLs. To get the tool `cfsign.pl`, go to Amazon CloudFront Signed URLs Helper Tool.

> **Note**
>
> Creating a URL signature is just one part of the process of serving private content using a signed URL. For more information about the entire process, see How to Serve Private Content Using a Signed URL (p. 73).

The following example shows how you might use `cfsign.pl` to create a streaming distribution signature.

```
$ cfsign.pl --action encode --stream example/video.mp4 --private-key
   /path/to/my-private-key.pem --key-pair-id PK12345EXAMPLE --expires 1265838202
```

This tool generates the policy statement from the command line arguments. The signature that it generates is an SHA1 hash of the policy statement that is then RSA encrypted.

The following is a sample Base64-encoded stream name.

```
mp4:example/video.mp4%3FPolicy%3DewogICJTdGF0ZW1lbnQiOlt7CiAgICAgICJSZXNvdXJjZSI
6ImRyciIsCiAgICAgICJDb25kaXRpb24iOnsKICAgICAgICAiSXBBBGRyZXNzIjp7IkFXUzpTb3VyY2V
JcCI6IjAuMC4wLjAvMCJ9LAogICAgICAgICJEYXRlTGVzc1RoYW4iOnsiQVdTOkVwb2NoVGltZSI6MjE
0NTkwNjIyMwMH0KICAgICAgfQogICAgEXAMPLE_%26Signature%3DewtHqEXK~68tsZt-eOFnZKGwTf2a
JlbKhXkK5SSiVqcG9pieCRV3xTEPtc29OzeXlsDvRycOM2WK0cXzcyYZhpl9tv2796ihHiCTAwIHQ8yP
17Af4nWtOLIZHoH6wkR3tU1cQHs8R1d-g-SlZGjNBXr~J2MbaJzm8i6EXAMPLE_%26Key-Pair-Id%3
DPK12345EXAMPLE
```

This signature authenticates the request to stream private content, `example/video.mp4`. If you're using Adobe Flash Player and the stream name is passed in from a web page using JavaScript, you must Base64-encode the signature and replace characters that are invalid in a URL request parameter (+, =, /) with characters that are valid (-, _, and ~, respectively). If the stream name is not passed in from a web page, you don't need to Base64-encode the signature. For example, you would not Base64-encode the

signature if you write your own player and the stream names are fetched from within the Adobe Flash .swf file.

The following example uses jwplayer with CloudFront.

```
<script type='text/javascript'>
  var so1 = new SWFObject
    ('http://d84l721fxaaqy9.cloudfront.net/player/player.swf',
    'mpl', '640', '360', '9');
  so1.addParam('allowfullscreen','true');
  so1.addParam('allowscriptaccess','always');
  so1.addParam('wmode','opaque');
  so1.addVariable('streamer','rtmp://s33r3xe4ayhhis.cloudfront.net/cfx/st');
  so1.addVariable("file","mp4:example/video.mp4%3FPolicy%3DewogICJTdGF0ZW1lbnQi

    Olt7CiAgICAgICJSZXNvdXJjZSI6ImRyciIsCiAgICAgICJDb25kaXRpb24iOnsKICAgICAgICA

    iSXBBZGRyZXNzIjp7IkFXUzpTb3VyY2VJcCI6IjAuMC4wLjAvMCJ9LAogICAgICAgICJEYXRlTG

    Vzc1RoYW4iOnsiQVdTOkVwb2NoVGltZSI6MjE0NTkxNjgwMH0KICAgICAgfQogICAgEXAMPLE_%

    26Signature%3DewtHqEXK~68tsZt-eOFnZKGwTf2aJlbKhXkK5SSiVqcG9pieCRV3xTEPtc29O

    zeXlsDvRycOM2WK0cXzcyYZhpl9tv2796ihHiCTAwIHQ8yP17Af4nWtOLIZHoH6wkR3tU1cQHs8

    R1d-g-SlZGjNBXr~J2MbaJzm8i6EXAMPLE_%26Key-Pair-Id%3DPK12345EXAMPLE
  so1.write('flv');
</script>
```

When you retrieve a stream to play from within an Adobe Flash .swf file, do not URL-encode the stream name, for example:

```
mp4:example/video.mp4?Policy=ewogICJTdGF0ZW1lbnQiOlt7CiAgICAgICJSZXNvdXJjZSI6ImR
yciIsCiAgICAgICJDb25kaXRpb24iOnsKICAgICAgICAiSXBBZGRyZXNzIjp7IkFXUzpTb3VyY2VJcCI
6IjAuMC4wLjAvMCJ9LAogICAgICAgICJEYXRlTGVzc1RoYW4iOnsiQVdTOkVwb2NoVGltZSI6MjE0NTk
xNjgwMH0KICAgICAgfQogICAgEXAMPLE_&Signature=ewtHqEXK~68tsZt-eOFnZKGwTf2aJlbKhXkK
5SSiVqcG9pieCRV3xTEPtc29OzeXlsDvRycOM2WK0cXzcyYZhpl9tv2796ihHiCTAwIHQ8yP17Af4nWt
OLIZHoH6wkR3tU1cQHs8R1d-g-SlZGjNBXr~J2MbaJzm8i6EXAMPLE_&Key-Pair-Id=PK12345
EXAMPLE
```

See the comments in the Perl source code for more information about the command line switches and features of this tool.

See also

# Create a URL Signature Using PHP

Any web server that runs PHP can use the PHP demo code to create policy statements and signatures for CloudFront streaming private distributions. The sample creates a functioning web page with signed

URL links that play a video stream using CloudFront streaming. To get the sample, download Signature Code for Video Streaming in PHP.

> 📋 **Note**
>
> Creating a URL signature is just one part of the process of serving private content using a signed URL. For more information about the entire process, see How to Serve Private Content Using a Signed URL (p. 73).

In the following code segment, the function `rsa_sha1_sign` hashes the policy and encrypts the result. The arguments required are a policy statement, an out parameter to contain the signature, and the private key for your AWS account or for a trusted AWS account that you specify. Next, the `url_safe_base64_encode` function creates a URL-safe version of the signature.

**Example RSA SHA1 Encryption in PHP**

```
function rsa_sha1_sign($policy, $private_key_filename) {
    $signature = "";

    // load the private key
    $fp = fopen($private_key_filename, "r");
    $priv_key = fread($fp, 8192);
    fclose($fp);
    $pkeyid = openssl_get_privatekey($priv_key);

    // compute signature
    openssl_sign($policy, $signature, $pkeyid);

    // free the key from memory
    openssl_free_key($pkeyid);

    return $signature;
}

function url_safe_base64_encode($value) {
    $encoded = base64_encode($value);
    // replace unsafe characters +, = and / with
    // the safe characters -, _ and ~
    return str_replace(
        array('+', '=', '/'),
        array('-', '_', '~'),
        $encoded);
}
```

The following code constructs a *canned* policy statement needed for creating the signature. For more information about canned policies, see Canned Policy (p. 86).

### Example Canned Signing Function in PHP

```
function get_canned_policy_stream_name($video_path, $private_key_filename,
$key_pair_id, $expires) {
    // this policy is well known by CloudFront, but you still need to sign it,

    // since it contains your parameters
    $canned_policy = '{"Statement":[{"Resource":"' . $video_path . '","Condi
tion":{"DateLessThan":{"AWS:EpochTime":'. $expires . '}}}]}';
    // the policy contains characters that cannot be part of a URL,
    // so we Base64 encode it
    $encoded_policy = url_safe_base64_encode($canned_policy);
    // sign the original policy, not the encoded version
    $signature = rsa_sha1_sign($canned_policy, $private_key_filename);
    // make the signature safe to be included in a url
    $encoded_signature = url_safe_base64_encode($signature);

    // combine the above into a stream name
    $stream_name = create_stream_name($video_path, null, $encoded_signature,
$key_pair_id, $expires);
    // url-encode the query string characters to work around a flash player bug

    return encode_query_params($stream_name);
    }
```

The following code constructs a *custom* policy statement needed for creating the signature. For more information about custom policies, see Canned Policy (p. 86).

### Example Custom Signing Function in PHP

```
function get_custom_policy_stream_name($video_path, $private_key_filename,
$key_pair_id, $policy) {
    // the policy contains characters that cannot be part of a URL, so we Base64
 encode it
    $encoded_policy = url_safe_base64_encode($policy);
    // sign the original policy, not the encoded version
    $signature = rsa_sha1_sign($policy, $private_key_filename);
    // make the signature safe to be included in a url
    $encoded_signature = url_safe_base64_encode($signature);

    // combine the above into a stream name
    $stream_name = create_stream_name($video_path, $encoded_policy, $encoded_sig
nature, $key_pair_id, null);
    // url-encode the query string characters to work around a flash player bug

    return encode_query_params($stream_name);

    }
```

For more information about OpenSSL implementation of RSA encryption, see The Open Source Toolkit for SSL/TLS.

See also

- GUI Tools for Signature Generation (p. 112)

# Create a URL Signature Using C# and the .NET Framework

The C# examples in this section implement a sample application that demonstrates how to create the signatures for CloudFront private distributions using canned and custom policy statements. The samples includes utility functions based on the AWS .NET SDK that can be useful in .NET applications.

> 📕 **Note**
>
> Creating a URL signature is just one part of the process of serving private content using a signed URL. For more information about the entire process, see How to Serve Private Content Using a Signed URL (p. 73).

To download the code, go to Signature Code in C#.

To use the RSA keys provided by AWS Account/Security in the .NET framework, you must convert the AWS-supplied .pem files to the XML format that the .NET framework uses. The OpenSSL Public and Private Key Parser available from .NET 2.0 OpenSSL Public and Private Key Parser will do the conversion.

After conversion, the RSA private key file is in the following format:

**Example RSA Private Key in the XML .NET Framework Format**

```
<RSAKeyValue>
  <Modulus>
    wO5IvYCP5UcoCKDo1dcspoMehWBZcyfs9QEzGi6Oe5y+ewGr1oW+vB2GPB
    ANBiVPcUHTFWhwaIBd3oglmF0lGQljP/jOfmXHUK2kUUnLnJp+oOBL2Ni
uFtqcW6h/L5lIpD8Yq+NRHg
    Ty4zDsyr2880MvXv88yEFURCkqEXAMPLE=
  </Modulus>
  <Exponent>AQAB</Exponent>
  <P>
    5bmKDaTz
    npENGVqz4Cea8XPH+sxt+2VaAwYnsarVUoS
BeVt8WLloVuZGG9IZYmH5KteXEu7fZveYd9UEXAMPLE==
  </P>
  <Q>
    1v9l/WN1a1N3rOK4VGoCokx7kR2SyTMSbZgF9IWJNOugR/WZw7HTnjipO3c9dy1Ms9pUKwUF4
    6d7049EXAMPLE==
  </Q>
  <DP>
    RgrSKuLWXMyBH+/l1Dx/I4tXuAJIrlPyo+VmiOc7b5NzHptkSHEPfR9s1
    OK0VqjknclqCJ3Ig86OMEtEXAMPLE==
  </DP>
  <DQ>
    pjPjvSFw+RoaTu0pgCA/jwW/FGyfN6iim1RFbkT4
    z49DZb2IM885f3vf35eLTaEYRYUHQgZtChNEV0TEXAMPLE==
  </DQ>
  <InverseQ>
    nkvOJTg5QtGNgWb9i
    cVtzrL/1pFEOHbJXwEJdU99N+7sMK+1066DL/HSBUCD63qD4USpnf0myc24in0EXAMPLE==</In
verseQ>
  <D>
      Bc7mp7XYHynuPZxChjWNJZIq+A73gm0ASDv6At7F8Vi9r0xUlQe/v0AQS3ycN8QlyR4XMbzMLYk

      3yjxFDXo4ZKQtOGzLGteCU2srANiLv26/imXA8FVidZftTAtLviWQZB
VPTeYIA69ATUYPEq0a5u5wjGy
      UOij9OWyuEXAMPLE=
  </D>
</RSAKeyValue>
```

The following C# code creates a signed URL that uses a canned policy by:

- Creating a policy statement.
- Hashing the policy statement using SHA1, and encrypting the result using RSA and the private key for your AWS account or for a trusted AWS account that you specify.
- Base64-encoding the hashed policy statement and replacing special characters to make the string safe to use as a URL request parameter.
- Concatenating the applicable values.

For the complete implementation, see the sample at Signature Code in C#.

**Example Canned Policy Signing Method in C#**

```csharp
public static string ToUrlSafeBase64String(byte[] bytes)
{
    return System.Convert.ToBase64String(bytes)
        .Replace('+', '-')
        .Replace('=', '_')
        .Replace('/', '~');
}

public static string CreateCannedPrivateURL(string urlString,
    string durationUnits, string durationNumber, string pathToPolicyStmnt,
    string pathToPrivateKey, string privateKeyId)
{
    // args[] 0-thisMethod, 1-resourceUrl, 2-seconds-minutes-hours-days
    // to expiration, 3-numberOfPreviousUnits, 4-pathToPolicyStmnt,
    // 5-pathToPrivateKey, 6-PrivateKeyId

    TimeSpan timeSpanInterval = GetDuration(durationUnits, durationNumber);

    // Create the policy statement.
    string strPolicy = CreatePolicyStatement(pathToPolicyStmnt,
        urlString,
        DateTime.Now,
        DateTime.Now.Add(timeSpanInterval),
        "0.0.0.0/0");
    if ("Error!" == strPolicy) return "Invalid time frame." +
        "Start time cannot be greater than end time.";

    // Copy the expiration time defined by policy statement.
    string strExpiration = CopyExpirationTimeFromPolicy(strPolicy);

    // Read the policy into a byte buffer.
    byte[] bufferPolicy = Encoding.ASCII.GetBytes(strPolicy);

    // Initialize the SHA1CryptoServiceProvider object and hash the policy data.

    using (SHA1CryptoServiceProvider
        cryptoSHA1 = new SHA1CryptoServiceProvider())
    {
        bufferPolicy = cryptoSHA1.ComputeHash(bufferPolicy);

        // Initialize the RSACryptoServiceProvider object.
        RSACryptoServiceProvider providerRSA = new RSACryptoServiceProvider();

        XmlDocument xmlPrivateKey = new XmlDocument();

        // Load the PrivateKey.xml file generated by ConvertPEMtoXML.
        xmlPrivateKey.Load(pathToPrivateKey);

        // Format the RSACryptoServiceProvider providerRSA and
        // create the signature.
        providerRSA.FromXmlString(xmlPrivateKey.InnerXml);
        RSAPKCS1SignatureFormatter rsaFormatter =
            new RSAPKCS1SignatureFormatter(providerRSA);
        rsaFormatter.SetHashAlgorithm("SHA1");
        byte[] signedPolicyHash = rsaFormatter.CreateSignature(bufferPolicy);
```

```
        // Convert the signed policy to URL-safe Base64 encoding and
        // replace unsafe characters + = / with the safe characters - _ ~
        string strSignedPolicy = ToUrlSafeBase64String(signedPolicyHash);

        // Concatenate the URL, the timestamp, the signature,
        // and the key pair ID to form the signed URL.
        return urlString +
            "?Expires=" +
            strExpiration +
            "&Signature=" +
            strSignedPolicy +
            "&Key-Pair-Id=" +
            privateKeyId;
    }
}
```

The following C# code creates a signed URL that uses a custom policy by:

- Creating a policy statement.
- Base64-encoding the policy statement and replacing special characters to make the string safe to use as a URL request parameter.
- Hashing the policy statement using SHA1, and encrypting the result using RSA and the private key for your AWS account or for a trusted AWS account that you specify.
- Base64-encoding the hashed policy statement and replacing special characters to make the string safe to use as a URL request parameter.
- Concatenating the applicable values.


For the complete implementation, see the sample at Signature Code in C#.

### Example Custom Policy Signing Method in C#

```
public static string ToUrlSafeBase64String(byte[] bytes)
{
    return System.Convert.ToBase64String(bytes)
        .Replace('+', '-')
        .Replace('=', '_')
        .Replace('/', '~');
}

public static string CreateCustomPrivateURL(string urlString,
    string durationUnits, string durationNumber, string startIntervalFromNow,

    string ipaddress, string pathToPolicyStmnt, string pathToPrivateKey,
    string PrivateKeyId)
{
    // args[] 0-thisMethod, 1-resourceUrl, 2-seconds-minutes-hours-days
    // to expiration, 3-numberOfPreviousUnits, 4-starttimeFromNow,
    // 5-ip_address, 6-pathToPolicyStmt, 7-pathToPrivateKey, 8-privateKeyId

    TimeSpan timeSpanInterval = GetDuration(durationUnits, durationNumber);
    TimeSpan timeSpanToStart = GetDurationByUnits(durationUnits,
        startIntervalFromNow);
    if (null == timeSpanToStart)
        return "Invalid duration units." +
            "Valid options: seconds, minutes, hours, or days";

    string strPolicy = CreatePolicyStatement(
        pathToPolicyStmnt, urlString, DateTime.Now.Add(timeSpanToStart),
        DateTime.Now.Add(timeSpanInterval), ipaddress);

    // Read the policy into a byte buffer.
    byte[] bufferPolicy = Encoding.ASCII.GetBytes(strPolicy);

    // Convert the policy statement to URL-safe Base64 encoding and
    // replace unsafe characters + = / with the safe characters - _ ~

    string urlSafePolicy = ToUrlSafeBase64String(bufferPolicy);

    // Initialize the SHA1CryptoServiceProvider object and hash the policy data.

    byte[] bufferPolicyHash;
    using (SHA1CryptoServiceProvider cryptoSHA1 =
        new SHA1CryptoServiceProvider())
    {
        bufferPolicyHash = cryptoSHA1.ComputeHash(bufferPolicy);

        // Initialize the RSACryptoServiceProvider object.
        RSACryptoServiceProvider providerRSA = new RSACryptoServiceProvider();

        XmlDocument xmlPrivateKey = new XmlDocument();

        // Load the PrivateKey.xml file generated by ConvertPEMtoXML.
        xmlPrivateKey.Load("PrivateKey.xml");

        // Format the RSACryptoServiceProvider providerRSA
        // and create the signature.
```

```
        providerRSA.FromXmlString(xmlPrivateKey.InnerXml);
        RSAPKCS1SignatureFormatter RSAFormatter =
            new RSAPKCS1SignatureFormatter(providerRSA);
        RSAFormatter.SetHashAlgorithm("SHA1");
        byte[] signedHash = RSAFormatter.CreateSignature(bufferPolicyHash);

        // Convert the signed policy to URL-safe Base64 encoding and
        // replace unsafe characters + = / with the safe characters - _ ~
        string strSignedPolicy = ToUrlSafeBase64String(signedHash);

        return urlString +
            "?Policy=" +
            urlSafePolicy +
            "&Signature=" +
            strSignedPolicy +
            "&Key-Pair-Id=" +
            PrivateKeyId;
    }
}
```

### Example Utility Methods for Signature Generation

The following methods get the policy statement from a file and parse time intervals for signature generation.

```csharp
public static string CreatePolicyStatement(string policyStmnt,
    string resourceUrl,
    DateTime startTime,
    DateTime endTime,
    string ipAddress)

{
    // Create the policy statement.
    FileStream streamPolicy = new FileStream(policyStmnt, FileMode.Open,
FileAccess.Read);
    using (StreamReader reader = new StreamReader(streamPolicy))
    {
        string strPolicy = reader.ReadToEnd();

        TimeSpan startTimeSpanFromNow = (startTime - DateTime.Now);
        TimeSpan endTimeSpanFromNow = (endTime - DateTime.Now);
        TimeSpan intervalStart =
            (DateTime.UtcNow.Add(startTimeSpanFromNow)) -
            new DateTime(1970, 1, 1, 0, 0, 0, DateTimeKind.Utc);
        TimeSpan intervalEnd =
            (DateTime.UtcNow.Add(endTimeSpanFromNow)) -
            new DateTime(1970, 1, 1, 0, 0, 0, DateTimeKind.Utc);

        int startTimestamp = (int)intervalStart.TotalSeconds; // START_TIME
        int endTimestamp = (int)intervalEnd.TotalSeconds;  // END_TIME

        if (startTimestamp > endTimestamp)
            return "Error!";

        // Replace variables in the policy statement.
        strPolicy = strPolicy.Replace("RESOURCE", resourceUrl);
        strPolicy = strPolicy.Replace("START_TIME", startTimestamp.ToString());

        strPolicy = strPolicy.Replace("END_TIME", endTimestamp.ToString());
        strPolicy = strPolicy.Replace("IP_ADDRESS", ipAddress);
        strPolicy = strPolicy.Replace("EXPIRES", endTimestamp.ToString());
        return strPolicy;
    }
}

public static TimeSpan GetDuration(string units, string numUnits)
{
    TimeSpan timeSpanInterval = new TimeSpan();
    switch (units)
    {
        case "seconds":
            timeSpanInterval = new TimeSpan(0, 0, 0, int.Parse(numUnits));
            break;
        case "minutes":
            timeSpanInterval = new TimeSpan(0, 0, int.Parse(numUnits), 0);
            break;
        case "hours":
            timeSpanInterval = new TimeSpan(0, int.Parse(numUnits), 0 ,0);
            break;
```

```
        case "days":
            timeSpanInterval = new TimeSpan(int.Parse(numUnits),0 ,0 ,0);
            break;
        default:
            Console.WriteLine("Invalid time units;" +
                "use seconds, minutes, hours, or days");
            break;
    }
    return timeSpanInterval;
}

private static TimeSpan GetDurationByUnits(string durationUnits,
    string startIntervalFromNow)
{
    switch (durationUnits)
    {
        case "seconds":
            return new TimeSpan(0, 0, int.Parse(startIntervalFromNow));
        case "minutes":
            return new TimeSpan(0, int.Parse(startIntervalFromNow), 0);
        case "hours":
            return new TimeSpan(int.Parse(startIntervalFromNow), 0, 0);
        case "days":
            return new TimeSpan(int.Parse(startIntervalFromNow), 0, 0, 0);
        default:
            return new TimeSpan(0, 0, 0, 0);
    }
}

public static string CopyExpirationTimeFromPolicy(string policyStatement)
{
    int startExpiration = policyStatement.IndexOf("EpochTime");
    string strExpirationRough = policyStatement.Substring(startExpiration +
        "EpochTime".Length);
    char[] digits = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9' };

    List<char> listDigits = new List<char>(digits);
    StringBuilder buildExpiration = new StringBuilder(20);

    foreach (char c in strExpirationRough)
    {
        if (listDigits.Contains(c))
            buildExpiration.Append(c);
    }
    return buildExpiration.ToString();
}
```

See also

# Create a URL Signature Using Java

The Open source Java toolkit for Amazon S3 and CloudFront provides sample code and information about CloudFront development in Java. For information about private distributions, go to Private Distributions at Programmer Guide: Code Samples.

> ### Note
>
> Creating a URL signature is just one part of the process of serving private content using a signed URL. For more information about the entire process, see How to Serve Private Content Using a Signed URL (p. 73).

The following methods are from the Java open source toolkit for Amazon S3 and CloudFront. You must convert the private key from PEM to DER format for Java implementations to use it.

### Example Java Policy and Signature Encryption Methods

```
// Signed URLs for a private distribution
// Note that Java only supports SSL certificates in DER format, so you will
need to
// convert your PEM-formatted file to DER format. To do this, you can use
openssl:
// openssl pkcs8 -topk8 -nocrypt -in origin.pem -inform PEM -out new.der -outform
 DER
// So the encoder works correctly, you should also add the bouncy castle jar
// to your project and then add the provider.

Security.addProvider(new org.bouncycastle.jce.provider.BouncyCastleProvider());

String distributionDomain = "a1b2c3d4e5f6g7.cloudfront.net";
String privateKeyFilePath = "/path/to/rsa-private-key.der";
String s3ObjectKey = "s3/object/key.txt";
String policyResourcePath = distributionDomain + "/" + s3ObjectKey;

// Convert your DER file into a byte array.

byte[] derPrivateKey = ServiceUtils.readInputStreamToBytes(new
    FileInputStream(privateKeyFilePath));

// Generate a "canned" signed URL to allow access to a
// specific distribution and object

String signedUrlCanned = CloudFrontService.signUrlCanned(
    "http://" + distributionDomain + "/" + s3ObjectKey, // Resource URL or Path

    keyPairId,       // Certificate identifier,
                     // an active trusted signer for the distribution
    derPrivateKey, // DER Private key data
    ServiceUtils.parseIso8601Date("2011-11-14T22:20:00.000Z") // DateLessThan
    );
System.out.println(signedUrlCanned);

// Build a policy document to define custom restrictions for a signed URL.

String policy = CloudFrontService.buildPolicyForSignedUrl(
    policyResourcePath, // Resource path (optional, may include '*' and '?'
wildcards)
    ServiceUtils.parseIso8601Date("2011-11-14T22:20:00.000Z"), // DateLessThan

    "0.0.0.0/0", // CIDR IP address restriction (optional, 0.0.0.0/0 means
everyone)
    ServiceUtils.parseIso8601Date("2011-10-16T06:31:56.000Z")  // DateGreaterThan
 (optional)
    );

// Generate a signed URL using a custom policy document.

String signedUrl = CloudFrontService.signUrl(
    "http://" + distributionDomain + "/" + s3ObjectKey, // Resource URL or Path

    keyPairId,       // Certificate identifier, an active trusted signer for the
 distribution
    derPrivateKey, // DER Private key data
```

```
    policy // Access control policy
    );
System.out.println(signedUrl);
```

See also

- Create a URL Signature Using Perl (p. 97)
- Create a URL Signature Using PHP (p. 98)
- Create a URL Signature Using C# and the .NET Framework (p. 101)
- GUI Tools for Signature Generation (p. 112)

# GUI Tools for Signature Generation

The following third-party tools provide interfaces that accept user input and create CloudFront private distributions and URL signatures.

| CloudBuddy | For information about using CloudBuddy for CloudFront private content, see Configuring CloudFront Distribution and Private Content. This tool is based on research at CSS CorpLabs for a .NET implementation of CloudFront private URLs. |
| --- | --- |
| Bucket Explorer | For information about using Bucket Explorer for CloudFront private content, see How to Create a Private Distribution on a Bucket. |
| CloudBerry | For information about using CloudBerry for CloudFront private content, see How to Configure Private Content for CloudFront Streaming with CloudBerry. |

See also

- Create a URL Signature Using Perl (p. 97)
- Create a URL Signature Using PHP (p. 98)
- Create a URL Signature Using C# and the .NET Framework (p. 101)
- Create a URL Signature Using Java (p. 109)

# Creating Secure HTTPS Connections

**Topics**

By default, CloudFront accepts both non-secure HTTP and secure HTTPS connections.

An HTTPS connection used in conjunction with a valid public key certificate (such as a certificate provided by VeriSign or DigiCert) validates your site identity and ensures that the data passed to and from your site is encrypted. However, HTTP connections don't ensure the identity of your site, and they don't provide data encryption.

## HTTPS Connections

You can use CloudFront to restrict access to your distributions to HTTPS connections. CloudFront passes end-user requests to your Amazon S3 bucket or custom origin in the same format it receives them. When you configure your distribution to refuse non-secure HTTP requests, CloudFront only passes HTTPS requests to Amazon S3 or to your custom origin. The concept is illustrated in the following graphic.
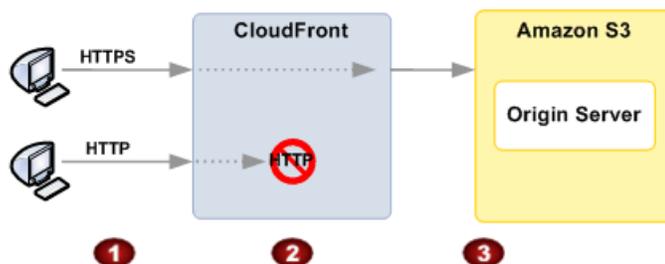
**Process for HTTP and HTTPS Requests When the Connection is Restricted**

| | |
|---|---|
| **1** | In the preceding graphic there are two different end users sending requests to CloudFront. One user sends an HTTPS request; the other user sends an HTTP request. |
| **2** | Because your distribution is configured to accept secure requests only, CloudFront refuses the non-secure HTTP request. |
| **3** | CloudFront passes the HTTPS request to Amazon S3. |

**STOP**   **Caution**

The only way to ensure that your end users retrieve an object using HTTPS is never to use any other protocol to fetch the object. If you have recently changed from HTTP to HTTPS, we recommend that you clear your objects' cache because cached objects are protocol agnostic. That means that an edge location will return an object from the cache irrespective of whether the current request protocol matches the protocol used previously. For information about expiring cached objects see Specifying When an Object Expires (p. 25).

# How to Restrict Access to Your Distribution to HTTPS Only

**Topics**

You can use either the AWS Management Console or the CloudFront API to restrict access to your distribution to HTTPS only. The following sections describe both methods.

## Using the AWS Management Console to Restrict Access

**To use the AWS Management Console to restrict access to your distribution**

1. Sign in to the AWS Management Console and open the Amazon CloudFront console at https://console.aws.amazon.com/cloudfront/.

2. Select the distribution to modify.

3. Click **Edit**.
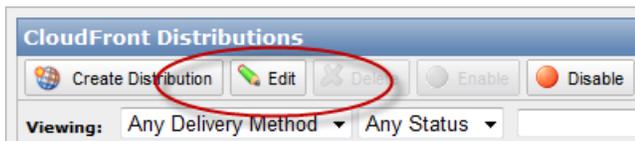


4. Under **Allowed Connections**, to restrict access to your distribution to HTTPS requests only, select **HTTPS Only**. (Selecting **HTTP and HTTPS** will allow CloudFront to use either protocol to serve the request, and will not limit allowed connections to HTTPS only.)
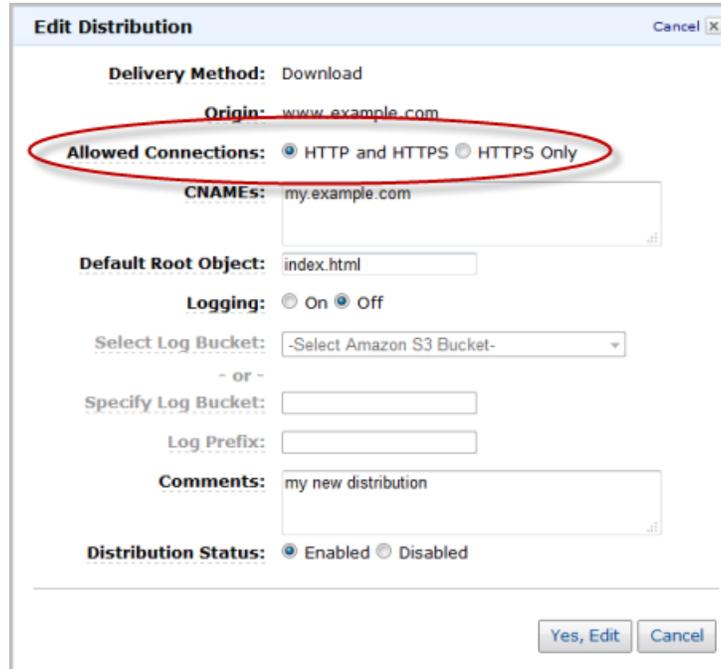
5.   To save your changes, click **Yes, Edit**.

> **Note**
>
> To create a new distribution with access restricted to HTTPS only, create a new distribution as
> described in the Amazon CloudFront Getting Started Guide, and select **HTTPS Only** when you
> specify **Allowed Connections**.

## Using the CloudFront API to Restrict Access

**To use the API to restrict access to your distribution**

1.   Create a new distribution (or update an existing distribution) so that it includes the
     `RequiredProtocols` element.
2.   Create public URLs for your distribution objects. For more information, see Public URLs for Objects
     in Amazon S3 (p. 22).

You don't need to change any configuration settings when using Amazon S3 as an origin.

The following example shows a CloudFront distribution's configuration with the `RequiredProtocols`
element.

```
<DistributionConfig xmlns="http://cloudfront.amazonaws.com/doc/2012-03-15/">
    <S3Origin>
        <DNSName>myawsbucket.s3.amazonaws.com</DNSName>
    </S3Origin>
    <CallerReference>20120229090000</CallerReference>
    <Comment>My comments</Comment>
    <Enabled>true</Enabled>
    <Logging>
```

```
      <Bucket>myawslogbucket.s3.amazonaws.com</Bucket>
      <Prefix>myprefix/</Prefix>
   </Logging>
   <RequiredProtocols>
     <Protocol>https</Protocol>
   </RequiredProtocols>
</DistributionConfig>
```

> ⚠ **Important**
>
> Currently, `https` is the only acceptable value for the `RequiredProtocols` element. Specifying no protocol or using a different value returns an error. For more information about using the `RequiredProtocols` element with the CloudFront control API, go to DistributionConfig Complex Type in the *Amazon CloudFront API Reference*.

If you want your distribution to accept both HTTPS and HTTP requests, simply omit the `RequiredProtocols` element. You can update your configuration at any time to include the `RequiredProtocols` element.

**Related Topics**

# CNAMEs and HTTPS

CloudFront doesn't support CNAMEs with HTTPS. If content is requested over HTTPS using CNAMEs, your end users' browsers will display the warning: *This page contains both secure and non-secure items.* To prevent this message from appearing, don't use CNAMEs with CloudFront HTTPS distributions.

# Charges for HTTPS Connections

You always incur a surcharge for HTTPS requests and bytes transferred. For information on billing rates, refer to the CloudFront pricing plan.

# Access Logs

**Topics**

Amazon CloudFront provides optional log files with information about end user access to your objects. This section describes how to enable and disable logging, the content of log files, and how AWS charges you if you decide to use logging.

### Note

If you use a custom origin, you will need to create an Amazon S3 bucket to store your log files in.

# Overview

You can enable CloudFront to deliver access logs per distribution to an Amazon S3 bucket of your choice. The following figure and table describe the basic process for access logs.
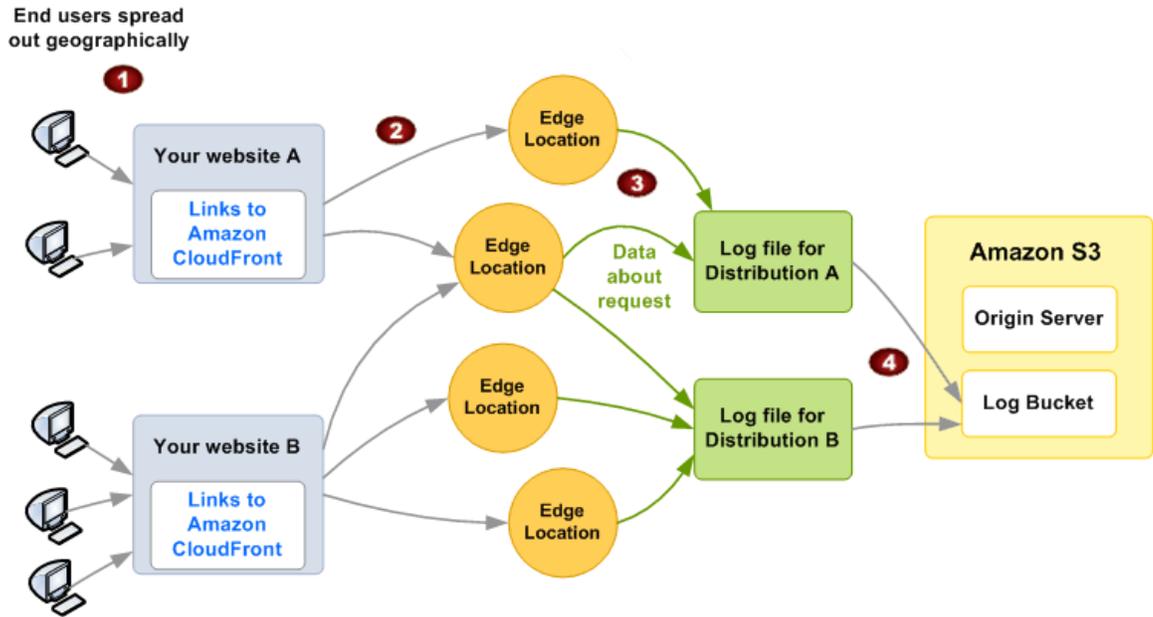
## Process for Access Logs

| | |
|---|---|
| **1** | Your end users use your application or website.<br><br>In this graphic, you have two different websites, A and B, each using a different CloudFront distribution (Distribution A and Distribution B). |
| **2** | Your end users send requests for content, and CloudFront routes each request to the appropriate edge location. |
| **3** | CloudFront writes data about each request to a log file specific to that distribution.<br><br>In this graphic, CloudFront writes information about requests related to Distribution A in a log file just for Distribution A, and requests for Distribution B in a log file just for Distribution B. |
| **4** | CloudFront periodically puts the distribution's log file in an Amazon S3 bucket of your choice, and then starts writing a new log file for the distribution. |

You can store each distribution's log files in the same bucket as your origin server or a different one. Each entry in a log file gives details about a single end user request for an object. You can have multiple distributions' log files delivered to the same bucket. When you enable logging for a particular distribution, you can specify an optional log filename prefix. Log files are delivered to your bucket within 24 hours of the end user's access, and typically sooner than that.

> **Note**
>
> Because logs for a single stream can get recorded in multiple files, we recommend you combine all the log files you receive for a given period into one file. You can then analyze the data for that period more quickly and accurately.

> **⚠ Important**
>
> You should use the logs to understand the nature of the requests for your content, not as a complete accounting of all requests. CloudFront delivers access logs on a best-effort basis. The log record for a particular request might be delivered long after the request was actually processed, or not at all. In rare cases, usage that appears in the AWS usage tracking and billing systems might not appear in CloudFront access logs.

# Bucket and File Ownership

You must have Amazon S3 `FULL_CONTROL` permission for the log file bucket. You have this permission by default if you're the bucket owner. If you're not, the bucket owner must grant your AWS account `FULL_CONTROL` permission.

When you enable logging, you do it with an API call to the CloudFront control API. Making that API call also automatically calls the Amazon S3 API to update the bucket's ACL to allow read and write permissions for the *AWS data feeds account*. This account writes the log files to the bucket.

Each log file has its own ACL (separate from the bucket's ACL). The bucket owner has `FULL_CONTROL` permission for the log files, the distribution owner (if not the bucket owner) has no permission, and the data feeds account has read and write permission.

> **📑 Note**
>
> Removing the permissions for the data feeds account does not disable logging. If you remove those permissions, but don't disable logging (which you do with the control API), we reinstate those permissions the next time the data feeds account needs to write a log file to your log bucket.

If you disable logging, we don't remove the read/write permissions for the data feeds account on either the bucket or the log files. If you want, you can do that yourself.

# How to Enable or Disable Logging

To enable or disable CloudFront access logs, you must use the 2009-04-02 or later version of the CloudFront control API for download distributions, and the 2010-05-01 or later version of the CloudFront control API for streaming distributions.

**To enable logging for a distribution**

1. Include a `Logging` element in the configuration object for a new or existing distribution.
2. Wait for the change to your configuration to take effect. The change might take up to 12 hours to take effect.
3. Send one or more requests to verify that logging is enabled.

> **📑 Note**
>
> If a distribution has no end user requests during a particular hour, you don't receive a log file for that hour.

For more information, see the following documentation:

- For information about the configuration object, see DistributionConfig Complex Type in the *Amazon CloudFront API Reference*.

- For information about including the `Logging` element in a new distribution, see POST Distribution in the *Amazon CloudFront API Reference.*

- For information about updating an existing distribution, see Updating a Distribution's Configuration (p. 53).

The `Logging` element includes two child elements: one for the Amazon S3 bucket to hold the logs, and one for the optional filename prefix of your choice.

> **Note**
>
> To enable easier listing of keys in a bucket, Amazon S3 users commonly use a prefix along with a slash (/) as a delimiter. CloudFront doesn't allow a prefix to begin with a slash; however, the prefix can end in one. The examples presented here have a slash appended following the prefix.

The following example shows a distribution's configuration with the `Logging` element.

```
<DistributionConfig xmlns="http://cloudfront.amazonaws.com/doc/2012-03-15/">
    <S3Origin>
        <DNSName>myawsbucket.s3.amazonaws.com</DNSName>
    </S3Origin>
    <CallerReference>20120229090000</CallerReference>
    <Comment>My comments</Comment>
    <Enabled>true</Enabled>
    <Logging>
        <Bucket>myawslogbucket.s3.amazonaws.com</Bucket>
        <Prefix>myprefix/</Prefix>
    </Logging>
</DistributionConfig>
```

You must specify the Amazon S3 bucket using this format: *<bucket name>*.s3.amazonaws.com. Do not use the Amazon S3 path style for specifying the bucket, which is s3.amazonaws.com/*<bucket name>*.

If you don't want to use a filename prefix, include an empty `Prefix` element, as shown in the following example. CloudFront doesn't substitute a default prefix. However, the XML is invalid if you omit the `Prefix` element entirely.

```
<DistributionConfig xmlns="http://cloudfront.amazonaws.com/doc/2012-03-15/">
    <S3Origin>
        <DNSName>myawsbucket.s3.amazonaws.com</DNSName>
    </S3Origin>
    <CallerReference>20120229090000</CallerReference>
    <Comment>My comments</Comment>
    <Enabled>true</Enabled>
    <Logging>
        <Bucket>myawslogbucket.s3.amazonaws.com</Bucket>
        <Prefix/>
    </Logging>
</DistributionConfig>
```

**To disable logging for a distribution**

- Remove the entire `Logging` element from the distribution's configuration.

```
<DistributionConfig xmlns="http://cloudfront.amazonaws.com/doc/2012-03-15/">

    <S3Origin>
        <DNSName>myawsbucket.s3.amazonaws.com</DNSName>
    </S3Origin>
    <CallerReference>20120229090000</CallerReference>
    <Comment>My comments</Comment>
    <Enabled>true</Enabled>
</DistributionConfig>
```

# How to Delete Log Files from an Amazon S3 Bucket

CloudFront does not automatically delete log files from the Amazon S3 bucket that you specified when you enabled logging. For information about deleting log files from an Amazon S3 bucket, see the applicable Amazon S3 documentation:

- Using the Amazon S3 console: See Deleting an Object in the *Amazon Simple Storage Service Console User Guide*.
- Using the REST API: See DELETE Object in the *Amazon Simple Storage Service API Reference*.
- Using the SOAP API: See DeleteObject in the *Amazon Simple Storage Service API Reference*.

# How to Change the Bucket or Prefix

At any time, you can update a distribution's logging configuration to use a different bucket or filename prefix. When you update the logging configuration, your changes take effect within 12 hours.

> ⚠️ **Important**
>
> Whenever you update the `Logging` element, you must provide both the `Bucket` and `Prefix` child elements. If you don't have a prefix, the `Prefix` element will be empty.

**To change the bucket or prefix**

1. Get the distribution's current configuration (for more information, go to GET Distribution Config in the *Amazon CloudFront API Reference*).
2. Update the `Logging` element with your desired changes.
3. Upload the new configuration (for more information, go to PUT Distribution Config in the *Amazon CloudFront API Reference*).

For more information about updating a distribution's configuration, see Updating a Distribution's Configuration (p. 53).

# File Naming and Timing of File Delivery

The filename follows this format (with the date and hour in UTC):

```
{Bucket}.s3.amazonaws.com/{Optional Prefix You Choose}{Distribution
ID}.{YYYY}-{MM}-{DD}-{HH}.{Unique ID}.gz
```

For example, if your bucket name is `mylogs`, and you name your prefix `myprefix/`, your filenames look similar to this:

```
mylogs.s3.amazonaws.com/myprefix/EMLARXS9EXAMPLE.2009-03-17-20.RT4KCN4SGK9.gz
```

Log files arrive in your bucket typically once an hour.

Each hour of usage is typically covered in a single log file. CloudFront compresses the file in gzip format before delivering it to your bucket. CloudFront might write multiple files for a given hour of usage. For example, this occurs if the log file contents for the hour exceed 50 MB (uncompressed).

> **Note**
>
> If a distribution has no end user requests during a particular hour, you don't receive a log file for that hour.

# Log File Format

The fields in download and streaming distribution log files are different. Both log files, however:

- Use the W3C extended log file format

  For more information, go to http://www.w3.org/TR/WD-logfile.html.
- Contain tab-separated values
- Contain records that are not necessarily in chronological order
- Contain two header lines: one with the file format version, and another that lists the W3C fields included in each record
- Substitutes URL encoded equivalents for spaces and non-standard characters in field values

  These non-standard characters consist of all ASCII codes below 32 and above 127, plus the characters in the following table. The URL encoding standard is RFC 1738. For more information, go to http://www.ietf.org/rfc/rfc1738.txt.

| Hexadecimal Value | Character |
|---|---|
| 0x3C | < |
| 0x3E | > |
| 0x22 | " |
| 0x23 | # |
| 0x25 | % |
| 0x7B | { |
| 0x7D | } |
| 0x7C | | |

| Hexadecimal Value | Character |
|---|---|
| 0x5C | \ |
| 0x5E | ^ |
| 0x7E | ~ |
| 0x5B | [ |
| 0x5D | ] |
| 0x60 | ` |
| 0x27 | ' |
| 0x20 | space |

# Download Distribution File Format

The following table describes the fields for one record in the download distribution log file.

| Field | Description |
|---|---|
| c-ip | Client IP, for example, 192.0.2.183. |
| cs(Host) | DNS name (the CloudFront distribution name specified in the request). If you made the request to a CNAME, the DNS name field will contain the underlying distribution DNS name, not the CNAME. |
| cs-method | HTTP access method. |
| cs(Referer) | The referrer. |
| cs(User-Agent) | The user agent. |
| date | The date (UTC) on which the event occurred, for example, 2009-03-10. |
| s-uri-stem | URI stem (e.g., /images/daily-ad.jpg). |
| sc-bytes | Server to client bytes, for example, 1045619. |
| sc-status | HTTP status code (e.g., 200). |
| time | Time when the server finished processing the request (UTC), for example, 01:42:39. |
| x-edge-location | The edge location that served the request. Each edge location is identified by a three-letter code and an arbitrarily assigned number, for example, DFW3. The three-letter code typically corresponds with the International Air Transport Association airport code for an airport near the edge location. (These abbreviations may change in the future.) For a list of edge locations, see the Amazon CloudFront detail page, http://aws.amazon.com/cloudfront. |
| cs-uri-query | The query string portion of the URI that is included on the connect string. When a URI doesn't contain a query string, the log file contains a single dash (-). The log records query strings to a maximum length of 8K bytes. The encoding standard is RFC 1738, as described in Log File Format (p. 122) |

> **Note**
>
> Question marks (?) in URLs and query strings are not included in the log.

The fields appear in the following order in a record:

- date
- time
- x-edge-location
- sc-bytes
- c-ip
- cs-method
- cs(Host)
- cs-uri-stem
- sc-status
- cs(Referer)
- cs(User Agent)
- cs-uri-query

The following is an example log file for a download distribution.

```
#Version: 1.0
#Fields: date time x-edge-location sc-bytes c-ip cs-method cs(Host) cs-uri-stem
 sc-status cs(Referer) cs(User-Agent) cs-uri-query
02/01/2011 01:13:11 FRA2 182 192.0.2.10 GET d2819bc28.cloudfront.net
/view/my/file.html 200 www.displaymyfiles.com Mozilla/4.0%20(compat
ible;%20MSIE%205.0b1;%20Mac_PowerPC) -
02/01/2011 01:13:12 LAX1 2390282 192.0.2.202 GET www.singalong.com
/soundtrack/happy.mp3 304 www.unknownsingers.com Mozilla/4.0%20(compat
ible;%20MSIE%207.0;%20Windows%20NT%205.1) a=b&c=d
```

# Streaming Distribution Log File Format

Each record in a streaming access log represents a playback event, for example, connect, play, pause, stop, disconnect, and so on. So, CloudFront generates multiple log records each time a viewer watches a video. To relate log records that stem from the same stream ID, use the *x-sid* field.

> **Note**
>
> Some fields are present for all events, whereas others appear only on Play, Stop, Pause, Unpause, and Seek events. When a field isn't relevant for a given event, the log file will contain a single dash (-).

The following table describes the fields that are present on each record in the streaming distribution log file, regardless of the type of event.

| Field | Description |
|-------|-------------|
| date | Date (UTC) on which the event occurred. |
| time | Time when the server received the request (UTC), for example, 01:42:39. |

| Field | Description |
|---|---|
| x-edge-location | The edge location where the playback event occured. Each edge location is identified by a three-letter code and an arbitrarily assigned number, for example, DFW3. The three-letter code typically corresponds with the International Air Transport Association airport code for an airport near the edge location. (These abbreviations may change in the future.) For a list of edge locations, see the Amazon CloudFront detail page, http://aws.amazon.com/cloudfront. |
| c-ip | Client IP, for example, 192.0.2.183. |
| x-event | The event type. This is a Connect, Disconnect, Play, Stop, Pause, Unpause, or Seek event. |
| sc-bytes | The running total number of bytes sent from the server to the client, up to the time of the event. |
| x-cf-status | A code indicating the status of the event. Currently, "OK" is the only value for this field. New functionality in the future could require new status codes. |
| x-cf-client-id | An opaque string identifier that can be used to differentiate clients
This value is unique for each connection. |
| cs-uri-stem | The stem portion of the URI, including the application and the application instance. This is sometimes referred to as the FMS *connect string*. For example, rtmp://shqshne4jdp4b6.cloudfront.net/cfx/st. |
| cs-uri-query | The query string portion of the URI that is included on the connect string. |
| c-referrer | The URI of the referrer. |
| x-page-url | The URL of the page from which the SWF is linked. |
| c-user-agent | The user agent. |

The following fields are present only on Play, Stop, Pause, Unpause, and Seek events. For other events, these fields will contain a single dash (-).

| Field | Description |
|---|---|
| x-sname | The stream name. |
| x-sname-query | The stream query string, if any. |
| x-file-ext | The stream type, for instance, FLV. |
| x-sid | The stream ID. This is a unique integer identifier for the connection. |

> **Note**
>
> Question marks (?) in URLs and query strings are not included in the log.

The fields appear in the following order in a record:

- date
- time
- x-edge-location

- c-ip
- x-event
- sc-bytes
- x-cf-status
- x-cf-client-id
- cs-uri-stem
- cs-uri-query
- c-referrer
- x-page-url
- c-user-agent
- x-sname
- x-sname-query
- x-file-ext
- x-sid

The following is an example of a log file for a streaming distribution.

```
#Version: 1.0
#Fields: date time x-edge-location c-ip x-event sc-bytes x-cf-status x-cf-client-
id cs-uri-stem cs-uri-query c-referrer x-page-url  c-user-agent x-sname x-sname-
query x-file-ext x-sid
2010-03-12  23:51:20  SEA4  192.0.2.147  connect  2014  OK
bfd8a98bee0840d9b871b7f6ade9908f  rtmp://shqshne4jdp4b6.cloudfront.net/cfx/st
  key=value  http://player.longtailvideo.com/player.swf  http://www.long
tailvideo.com/support/jw-player-setup-wizard?example=204  LNX%2010,0,32,18
-  -  -  -
2010-03-12  23:51:21  SEA4  192.0.2.222  play  3914  OK
bfd8a98bee0840d9b871b7f6ade9908f  rtmp://shqshne4jdp4b6.cloudfront.net/cfx/st
  key=value  http://player.longtailvideo.com/player.swf  http://www.long
tailvideo.com/support/jw-player-setup-wizard?example=204  LNX%2010,0,32,18
myvideo  p=2&q=4  flv  1
2010-03-12  23:53:44  SEA4  192.0.2.4  stop  323914  OK
bfd8a98bee0840d9b871b7f6ade9908f  rtmp://shqshne4jdp4b6.cloudfront.net/cfx/st
  key=value  http://player.longtailvideo.com/player.swf  http://www.long
tailvideo.com/support/jw-player-setup-wizard?example=204  LNX%2010,0,32,18
dir/other/myvideo  p=2&q=4  flv  1
2010-03-12  23:53:44  SEA4  192.0.2.103  play  8783724  OK
bfd8a98bee0840d9b871b7f6ade9908f  rtmp://shqshne4jdp4b6.cloudfront.net/cfx/st
  key=value  http://player.longtailvideo.com/player.swf  http://www.long
tailvideo.com/support/jw-player-setup-wizard?example=204  LNX%2010,0,32,18
dir/favs/myothervideo  p=42&q=14  mp4  2
2010-03-12  23:56:21  SEA4  192.0.2.199  stop  429822014  OK
bfd8a98bee0840d9b871b7f6ade9908f  rtmp://shqshne4jdp4b6.cloudfront.net/cfx/st
  key=value  http://player.longtailvideo.com/player.swf  http://www.long
tailvideo.com/support/jw-player-setup-wizard?example=204  LNX%2010,0,32,18
dir/favs/myothervideo  p=42&q=14  mp4  2
2010-03-12  23:59:44  SEA4  192.0.2.14  disconnect  429824092  OK
bfd8a98bee0840d9b871b7f6ade9908f  rtmp://shqshne4jdp4b6.cloudfront.net/cfx/st
  key=value  http://player.longtailvideo.com/player.swf  http://www.long
tailvideo.com/support/jw-player-setup-wizard?example=204  LNX%2010,0,32,18
-  -  -  -
```

# Charges for Access Logs

Access logging is an optional feature of CloudFront. There is no extra charge for enabling access logging. However, you accrue the usual Amazon S3 charges for storing and accessing the files on Amazon S3 (you can delete them at any time).

**Related Topics**

- Paying for CloudFront (p. 11)

# General Usage Data

In addition to the optional CloudFront access logs (for more information, see Access Logs (p. 117)), you can get other, more general information about your AWS service usage at no cost.

## AWS Account Activity

You can get general information about your AWS service usage and costs on your Account Activity page.

**To access your account activity**

- Go to http://aws.amazon.com, click **Your Account**, and then select **Account Activity**.

## Usage Report

AWS provides a usage report for Amazon CloudFront, similar to the one for Amazon S3. You can download aggregate usage by edge location region (Europe, U.S., Japan, Asia-Pacific) and by usage type (data transferred out and requests). You can aggregate the data by hour, day, or month. If you want information about the GETS from your origin server, refer to the usage report for Amazon S3.

**To get usage reports**

- Go to http://aws.amazon.com, click **Your Account**, and then select **Usage Reports**.

> **Note**
>
> The value for the `Resource` field in the CloudFront usage report is your distribution's ID.

# Making API Requests

**Topics**

This section describes how to make REST requests to the Amazon CloudFront *control API*, which you use to create and manage your distributions. The various topics acquaint you with the components of requests, the content of responses, and how to authenticate requests.

# Endpoints

Unlike other Amazon services, CloudFront doesn't have multiple endpoints based on the regions in which the service operates (e.g., Singapore, EU/Dublin, US/East, and so on). This is because CloudFront distributions aren't regional resources like Amazon S3 buckets and Amazon EC2 instances. Instead, Amazon CloudFront has the ability to serve content from one of its many edge locations. This means that CloudFront distributions have a single endpoint: the location of the origin server for a specific distribution.

As a result, when you make a REST request you use the following format, where *<distribution>* is the distribution that you are asking to take action on in your request.

```
cloudfront.amazonaws.com/2012-03-15/<distribution>
```

**Related Topics**

- REST Requests (p. 130)

- The Amazon CloudFront Network (a list on the AWS website of all the Amazon CloudFront edge locations)

- Regions and Endpoints (information about AWS product endpoints and regions in the Amazon Web Services General Reference)

# AWS Support for Programming Languages

AWS provides libraries, sample code, tutorials, and other resources for software developers who prefer to build applications using language-specific APIs instead of CloudFront's REST API. These libraries provide basic functions (not included in CloudFront's REST API), such as request authentication, request retries, and error handling so you can get started more easily. Libraries and resources are available for the following languages:

- Java
- PHP
- Ruby
- Windows and .NET

For libraries and sample code in all languages, go to Sample Code & Libraries.

# REST Requests

Amazon CloudFront REST requests are HTTPS requests, as defined by RFC 2616 (for more information, go to http://www.ietf.org/rfc/rfc2616.txt). This section describes the structure of a CloudFront REST request. For detailed descriptions of the actions you can perform, go to the Amazon CloudFront API Reference.

A typical REST action consists of sending a single HTTPS request to CloudFront, and waiting for the HTTP response. Like any HTTP request, a REST request to CloudFront contains a request method, a URI, request headers, and sometimes a query string or request body. The response contains an HTTP status code, response headers, and sometimes a response body.

## Request URI

The request URI always starts with a forward slash and then the version of the CloudFront API you use (for example, 2012-03-15). The remainder of the URI indicates the particular resource you want to act on. For example, following is the URI you use when creating a new distribution (for more information, go to POST Distribution in the *Amazon CloudFront API Reference*).

```
/2012-03-15/distribution
```

The topics in this guide that describe the different API actions show how to structure the URI.

## Request Headers

The following table lists the HTTP headers that CloudFront REST requests use.

| Header Name | Description | Required |
|---|---|---|
| Authorization | The information required for request authentication (). | Yes |
| Content-Length | Length of the message (without the headers) according to RFC 2616.<br>Condition: Required if the request body itself contains information (most toolkits add this header automatically). | Conditional |

| Header Name | Description | Required |
|---|---|---|
| `Content-Type` | The content type of the resource. Example: `text/plain`.<br>Condition: Required for POST and PUT requests. | Conditional |
| `Date` | The date used to create the signature contained in the `Authorization` header. The format must be one of the full date formats specified in RFC 2616 section 3.1.1. For example: `Wed, 05 Apr 2006 21:12:00 GMT`. For more information, go to the RFC 2616 specification.<br>Condition: Required unless you provide the `x-amz-date` header (for more information about the request time stamp, ). | Conditional |
| `Host` | The host being requested. The value must be `cloudfront.amazonaws.com`<br>Condition: Required for HTTP 1.1 (most toolkits add this header automatically) | Conditional |
| `x-amz-date` | The date used to create the signature contained in the `Authorization` header. The format must be one of the full date formats specified in RFC 2616 section 3.1.1. For example: `Wed, 05 Apr 2006 21:12:00 GMT`. For more information, go to the RFC 2616 specification.<br>Condition: Required if you do not provide the `Date` header (for more information, ). | Conditional |

# Request Time Stamp

You must provide the time stamp in either the HTTP `Date` header or the AWS `x-amz-date` header (some HTTP client libraries don't let you set the `Date` header). When an `x-amz-date` header is present, the system ignores any `Date` header when authenticating the request.

The time stamp must be within 15 minutes of the AWS system time when the request is received. If it isn't, the request fails with the `RequestExpired` error code. This is to prevent replays of your requests by an adversary.

# Request Body

Many of the CloudFront API actions require you to include XML in the body of the request. The XML conforms to the CloudFront schema. The topics in this guide that describe the API actions show the structure of the XML required in the request.

# Example Request

The following example request creates a distribution in the CloudFront system.

```
POST /2012-03-15/distribution HTTP/1.1
Host: cloudfront.amazonaws.com
Authorization: [AWS authentication string]
Date: Thu, 19 Nov 2009 19:37:58 GMT
[Other required headers]
```

```
<?xml version="1.0" encoding="UTF-8"?>
<DistributionConfig xmlns="http://cloudfront.amazonaws.com/doc/2012-03-15/">
    <S3Origin>
        <DNSName>myawsbucket.s3.amazonaws.com</DNSName>
    </S3Origin>
    <CallerReference>20120229090000</CallerReference>
    <Comment>My comments</Comment>
    <Enabled>true</Enabled>
</DistributionConfig>
```

## Related Topics

- Authenticating REST Requests (p. 135)
- REST Responses (p. 133)

# REST Responses

Amazon CloudFront responses are just standard HTTP responses. Some of the CloudFront actions return special information specific to CloudFront in the form of an HTTP header or XML in the body of the response. The specific details are covered in the API reference topic for the particular action.

## Request ID

Each response contains a request ID that you can use if you need to troubleshoot a request with AWS. The ID is contained in an HTTP header called `x-amz-request-id`. An example of a request ID is `647cd254-e0d1-44a9-af61-1d6d86ea6b77`.

## Example Response

The following example shows the response when creating a distribution.

```
201 Created
Location: https://cloudfront.amazonaws.com/2012-03-15/distribution/EDFDVBD6EX
AMPLE
x-amz-request-id: request_id


<?xml version="1.0" encoding="UTF-8"?>
<Distribution xmlns="http://cloudfront.amazonaws.com/doc/2012-03-15/">
   <Id>EDFDVBD6EXAMPLE</Id>
   <Status>InProgress</Status>
   <LastModifiedTime>2009-11-19T19:37:58Z</LastModifiedTime>
   <DomainName>d111111abcdef8.cloudfront.net</DomainName>
   <DistributionConfig>
      <S3Origin>
          <DNSName>myawsbucket.s3.amazonaws.com</DNSName>
      </S3Origin>
      <CallerReference>20120229090000</CallerReference>
      <Comment>My comments</Comment>
      <Enabled>true</Enabled>
   </DistributionConfig>
</Distribution>
```

## Error Responses

If a REST request results in an error, the HTTP reply has:

* An XML error document as the response body
* Content-Type header: application/xml
* An appropriate 3xx, 4xx, or 5xx HTTP status code

Following is an example of the XML error document in a REST error response.

```
<ErrorResponse xmlns="http://cloudfront.amazonaws.com/doc/2012-03-15/">
   <Error>
      <Type>Sender</Type>
      <Code>InvalidURI</Code>
```

```
      <Message>Could not parse the specified URI.</Message>
   </Error>
   <RequestId>410c2a4b-e435-49c9-8382-3770d80d7d4c</RequestId>
</ErrorResponse>
```

## Related Topics

- Errors (in the *Amazon CloudFront API Reference*)
- REST Requests (p. 130)
- Authenticating REST Requests (p. 135)

# Authenticating REST Requests

**Topics**

Every request you make to the Amazon CloudFront control API must be authenticated. AWS and others in the coding community provide tools that automatically sign your requests as required for CloudFront. For more information, go to the Amazon CloudFront Getting Started Guide or to the CloudFront sample code and libraries page. If you plan to write your own code to sign requests, then read this topic.

## Comparison with Amazon S3

If you already know how authentication works for Amazon Simple Storage Service REST requests, then the information in this topic will be familiar to you.

Following are the main differences between how you authenticate CloudFront and Amazon S3 requests:

- For CloudFront, you must use HTTPS
- For CloudFront, the canonical string to sign is simply the value of the `Date` header (or the `x-amz-date` header if you include it in the request)
  Therefore, the value of the `Authorization` header is as follows:

```
Authorization: "AWS" + " " + AWSAccessKeyID + ":" +
               Base64(HMAC-SHA1(UTF-8(Date), UTF-8(AWSSecretAccessKey)))
```

## Overview of the Authentication Process

Authentication is how you prove your identity to the system. You must prove your identity in all your requests to the CloudFront control API. The following sections describe how.

The CloudFront REST API uses a custom HTTP scheme based on a keyed-HMAC (Hash Message Authentication Code) for authentication. The following figure and table describe the basic process for authentication.
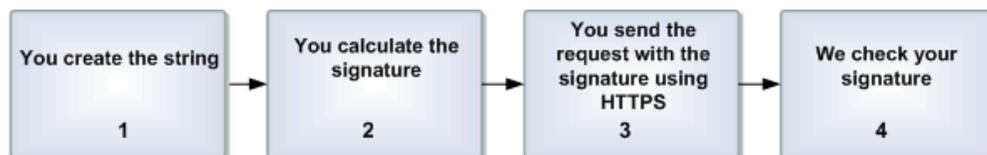


**Process for Request Authentication**

| 1 | You create a string based on specific information in the request. For more information, see The String to Sign (p. 136). |
|---|---|

| 2 | You calculate a *signature* using your AWS Secret Access Key, the string from task 1, and an HMAC-SHA1 algorithm. |
| | Informally, we call this process *signing the request*, and we call the output of the HMAC algorithm the signature because it simulates the security properties of a real signature. For instructions on creating the signature, see Calculating the Signature (p. 136). |
| 3 | You include the signature in the request and send the request to AWS using HTTPS (HTTP requests are not accepted). |
| | For information about where to put the signature in the request, see The Authorization Header (p. 137). |
| 4 | We check your signature. |
| | When we receive your request, we fetch the AWS Secret Access Key that you claim to have and use it in the same way you did to compute a signature for the message. We then compare the signature we calculated to the signature you presented in the request. If the two signatures match, we accept and process the request. Otherwise, we reject the request and respond with an error message (for more information, see Authentication Errors (p. 137)). |

> **Note**
>
> We also confirm the request time stamp is within 15 minutes of the AWS server time. For more information, see Fetching the Date (p. 138).

# The String to Sign

In the first task in the preceding process, you form a string. The string is simply the UTF-8 encoded value of the `Date` header in the request (e.g., Thu, 19 Nov 2009 19:37:58 GMT). Your request must include either the `Date` header, the `x-amz-date` header, or both (if both are present, we ignore the `Date` header when authenticating the request). You might decide to include the `x-amz-date` header if your HTTP client doesn't let you set the `Date` header.

The format you use for the header value must be one of the full date formats specified in RFC 2616, section 3.3.1, for example, `Wed, 05 Apr 2006 21:12:00 GMT`. For more information, go to the RFC 2616 specification.

# Calculating the Signature

Calculating the value to include in the request is a simple procedure.

**Calculating the Signature**

1. Calculate an RFC 2104-compliant HMAC-SHA1 hash, using the string (see The String to Sign (p. 136)) and your Secret Access Key as the key.
2. Convert the resulting value to base64.

   The result is the signature you include in the request.

The following table shows a string, a fake Secret Access Key, and what the resulting base64 encoded signature would be.

| String | `Thu, 14 Aug 2008 17:08:48 GMT` |

| Secret Access Key | `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY` |
|---|---|
| Base64 encoded signature | `4cP3hCesdCQJ1jP11111YSu0g=EXAMPLE` |

# The Authorization Header

To pass the signature to AWS, you include it as part of the standard HTTP `Authorization` header. You include both the signature and your AWS Access Key ID in the header using the following format.

```
Authorization: AWS <AWSAccessKeyId>:<Signature>
```

Note that there is a space after the `AWS`.

Following is an example REST request with the example signature calculated in the preceding section. The AWS Access Key ID (`AKIAIOSFODNN7EXAMPLE`) is fake.

```
POST /2012-03-15/distribution HTTP/1.1
Host: cloudfront.amazonaws.com
Date: Thu, 14 Aug 2008 17:08:48 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:111111112222222jPXo7+e/YSu0g=
[Other required headers]

<?xml version="1.0" encoding="UTF-8"?>
<DistributionConfig xmlns="http://cloudfront.amazonaws.com/doc/2012-03-15/">
   <S3Origin>
      <DNSName>myawsbucket.s3.amazonaws.com</DNSName>
   </S3Origin>
   <CallerReference>20120229090000</CallerReference>
   <Comment>My comments</Comment>
   <Enabled>true</Enabled>
</DistributionConfig>
```

# Authentication Errors

If the signature we create based on your request and Secret Access Key doesn't match the signature you sent in the request, we return the following error.

```
<ErrorResponse xmlns="http://cloudfront.amazonaws.com/doc/2012-03-15/">
   <Error>
      <Type>Sender</Type>
      <Code>SignatureDoesNotMatch</Code>
      <Message>The request signature we calculated
        does not match the signature you provided.
        Check your AWS Secret Access Key and signing
        method. Consult the service documentation for details.
      </Message>
   </Error>
   <RequestId>a1170c87-d04d-47c9-964f-54e1a4883f4e</RequestId>
</ErrorResponse>
```

# Fetching the Date

To avoid replays of your requests, AWS requires the time stamp in the request to be within 15 minutes of the AWS system time. To avoid clock synchronization errors, we recommend you fetch the current date from the CloudFront server and then use that as the time stamp for your request and the string for your signature.

**To fetch the date**

- Send an unauthenticated GET request for the date resource.

```
GET /date HTTP/1.1
Host: cloudfront.amazonaws.com
```

We return the current server date as the value of the `Date` response header (note that the HTTP status code may or may not be a 200). The date uses the RFC 1123 format (e.g., Wed, 18 Nov 2009 17:08:48 GMT). For more information, go to the RFC 1123 specification.

# CloudFront Tutorials

The following tutorials explain how to use CloudFront for live streaming and for geoblocking:

# Live HTTP Streaming Using CloudFront and Adobe Flash Media Server 4.5

**Topics**

## Overview of Live HTTP Streaming with Amazon Web Services

Live streaming with Amazon Web Services allows you to use the features of Adobe Flash Media Server version 4.5, including live video streaming where your live video is delivered by a series of HTTP requests

from the player that is controlled by manifest files. Flash Media Server 4.5 supports two HTTP file formats: HLS (HTTP Live Streaming) for iOS devices and HDS (HTTP Dynamic Streaming) for Flash applications. You can stream high-quality media using the free Flash Media Live Encoder desktop application either for Windows or for Mac OS.

Video is broken up into a series of smaller files (called segments or fragments) that are cached in the CloudFront network for improved performance. The live streams are then delivered to both Flash Player–compatible and iOS devices through standard HTTP caching and network infrastructures using the integrated HTTP server as an origin service in Flash Media Server 4.5. To review the new capabilities of Adobe Flash Media Server 4.5, go to What's New in Flash Media Server 4.5 on the Adobe website.

To set up live streaming with Amazon Web Services (AWS), review the system requirements for Adobe Flash Player at http://www.adobe.com/products/flashplayer/systemreqs/. Then perform the procedures in the following sections:

1. Creating an Amazon Web Services Account (p. 140)
2. Creating an Amazon EC2 Key Pair (p. 141)
3. Subscribing to Adobe Flash Media Server (p. 142)
4. Creating an AWS CloudFormation Stack for Live Streaming (p. 143)
5. Verifying that Adobe Flash Media Server Is Running (p. 147)
6. Setting Up Adobe Flash Media Live Encoder to Publish a Live Stream (p. 148)
7. Embedding Flash Media Playback for an Amazon CloudFront Live HTTP Stream in a Web Application (p. 151)
8. Deleting an AWS CloudFormation Stack for Live Streaming (p. 153)

For frequently asked questions, see Frequently Asked Questions (p. 154).

For links to additional Adobe and AWS documentation, see Additional Documentation (p. 159).

For information about the Adobe Flash Media Server features available on Amazon Web Services, see http://www.adobe.com/products/flashmediaserver/amazonwebservices.

# Creating an Amazon Web Services Account

If you already have an AWS account, skip to Creating an Amazon EC2 Key Pair (p. 141). If you don't already have an AWS account, use the following procedure to create one.

> **Note**
>
> When you create an account, AWS automatically signs up the account for all services. You are charged only for the services you use.

**To create an AWS account**

1. Go to http://aws.amazon.com, and click **Create an AWS Account**.
2. Follow the on-screen instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.

Next: Creating an Amazon EC2 Key Pair (p. 141)

# Creating an Amazon EC2 Key Pair

If you already have an Amazon EC2 key pair in the Amazon EC2 region in which you want to configure live streaming, skip to Subscribing to Adobe Flash Media Server (p. 142). If you don't have a key pair in that region, perform the following procedure.

A key pair is a security credential similar to a password. You specify a key pair when you create an AWS CloudFormation stack for live streaming, later in this process. After live streaming is configured, you use the key pair to securely connect to your Amazon EC2 instance.

### To create an Amazon EC2 key pair

1.  Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

2.  In the Navigation pane, in the Region list, click the region in which you want to create the key pair.
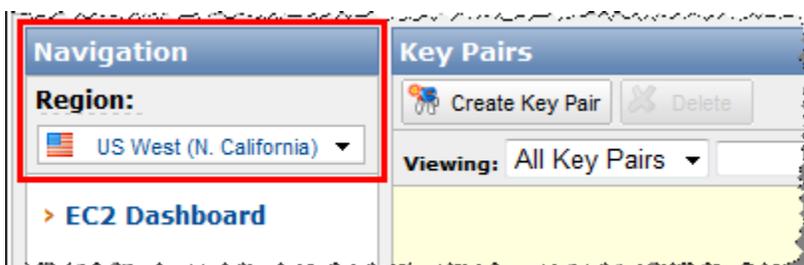
    You must create the key pair in the same region where you will create your AWS CloudFormation stack for live streaming later in this process. We recommend that you create the key pair and the stack for live streaming in the region that is closest to the users who will be doing the streaming.

3.  In the Navigation pane, click **Key Pairs**.
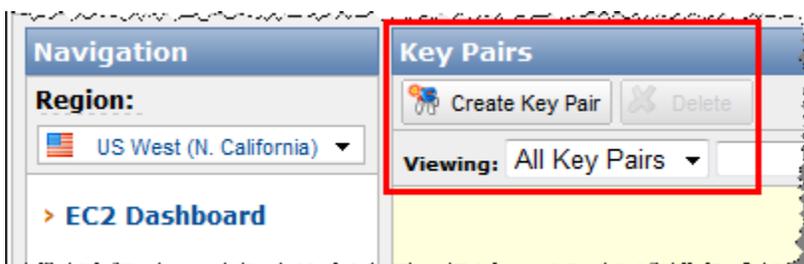
4.  In the Key Pairs pane, click **Create Key Pair**.

5.  In the Create Key Pair dialog box, enter a name for the key pair, and make note of the name. You'll enter this value when you create an AWS CloudFormation live-streaming stack, later in the process of setting up live streaming.

6.  Click **Create**.

7.  In the Opening <key_pair_name>.pem dialog box, save the .pem file to a safe place on your computer.

8.  Click **Close** to close the Create Key Pair dialog box.

Next: Subscribing to Adobe Flash Media Server (p. 142)

# Subscribing to Adobe Flash Media Server

Perform the following procedure to subscribe to Adobe Flash Media Server for Amazon Web Services with your AWS account.

There is a $5.00 monthly subscription fee. This fee allows you to run an unlimited number of Flash Media Server instances. In addition to the monthly subscription fee, there is a fee for hourly usage and a fee for data transfer. For more information, see Amazon EC2 Pricing at http://aws.amazon.com/ec2/pricing.

**To order Adobe Flash Media Server for Amazon Web Services**

1.  Go to http://www.adobe.com/go/learn_fms_aws_order_en.
2.  Follow the on-screen instructions.

    **Note**

    Read the product license agreement at  http://www.adobe.com/products/eulas.

3.  Read the pricing terms, and click **Place Your Order**.

# Creating an AWS CloudFormation Stack for Live Streaming

The following procedure uses an AWS CloudFormation template to create a stack that launches the AWS resources required by live streaming.

**To create an AWS CloudFormation stack for live streaming**

1. In the following list, click the Amazon EC2 Region where you want to create the stack. The Create Stack wizard starts, and a region-specific value is automatically entered in the **Provide a Template URL** field.

   US East (Virginia)

   US West (Oregon)

   US West (Northern California)

   EU (Ireland)

   Asia Pacific (Singapore)

   Asia Pacific (Tokyo)

   South America (Sao Paulo)

   > **Note**
   >
   > If you want users to view your live stream using a Flash-based player that is hosted on your own domain, see How do I update crossdomain.xml so my users can view the live stream using a Flash-based player that is hosted on my own domain? (p. 155).

2. If you are not already signed in to the AWS Management Console, sign in when prompted.
3. *Optional:* Change the value of the **Stack Name** field. The stack name must not contain spaces, and it must be unique within your AWS account.

4. Do not change the **Stack Template Source** option or the value of **Provide a Template URL**.

5. *Optional:* To configure SNS notification, to specify how long you're willing to wait for the stack to be created, and to choose whether to roll back changes if stack creation fails, check the **Show Advanced Options** checkbox, and specify the applicable values.

6. Click **Continue**, and the Specify Parameters page of the Create Stack wizard appears.

7. In the **KeyPair** field, enter the name of an Amazon EC2 key pair in the region in which you want to create the stack for live streaming. The key pair must be associated with the account that you're currently logged on with. If you created a key pair when you performed the procedure in Creating an Amazon EC2 Key Pair (p. 141), enter the name of that key pair.

8. In the **StreamName** field, enter a short name (without spaces) for your live stream.

9. In the **FMSAdminUserName** field, enter the user name that you want to use to log on to the Flash Media Administration Console after your Amazon EC2 Flash Media Server instance is created.

10. In the **FMSAdminPassword** field, enter a password (minimum 8 characters) for the user name that you specified in Step 9.

11. In the **InstanceType** field, enter an instance type, and click **Continue**. The default value is *m1.xlarge*.

    The instance type determines pricing for your Adobe Flash Media Server instance. The following table lists the Amazon EC2 instance types that are supported for live streaming.

    For more information about Amazon EC2 instance types, see Available Instance Types.

    For information about pricing, go to the Adobe Flash Media Server on Amazon Web Services page, http://www.adobe.com/ap/products/flashmediaserver/amazonwebservices/, and click the **Pricing** tab.

| Amazon EC2 and Adobe Flash Media Server Instance Types | InstanceType Code |
|---|---|
| Large | m1.large |
| High-memory extra large | m2.xlarge |
| High-CPU extra large | c1.xlarge |

| Amazon EC2 and Adobe Flash Media Server Instance Types | InstanceType Code |
|---|---|
| Extra large | m1.xlarge |
| High-memory double extra large | m2.2xlarge |
| High-memory quadruple extra large | m2.4xlarge |

12. Review the settings for the stack. When you're satisfied with the settings, click **Create Stack**.

Your stack may take several minutes to create. To track the progress of the stack creation, select the stack, and click the **Events** tab in the bottom frame. If AWS CloudFormation cannot create the stack, the Events tab lists error messages.

When your stack is ready, in the top frame, the status for the stack changes to **CREATE_COMPLETE**.



When your stack is created, click the **Outputs** tab, which displays the stack creation outputs. You will use these values when you set up Adobe Flash Media Live Encoder, later in the process.
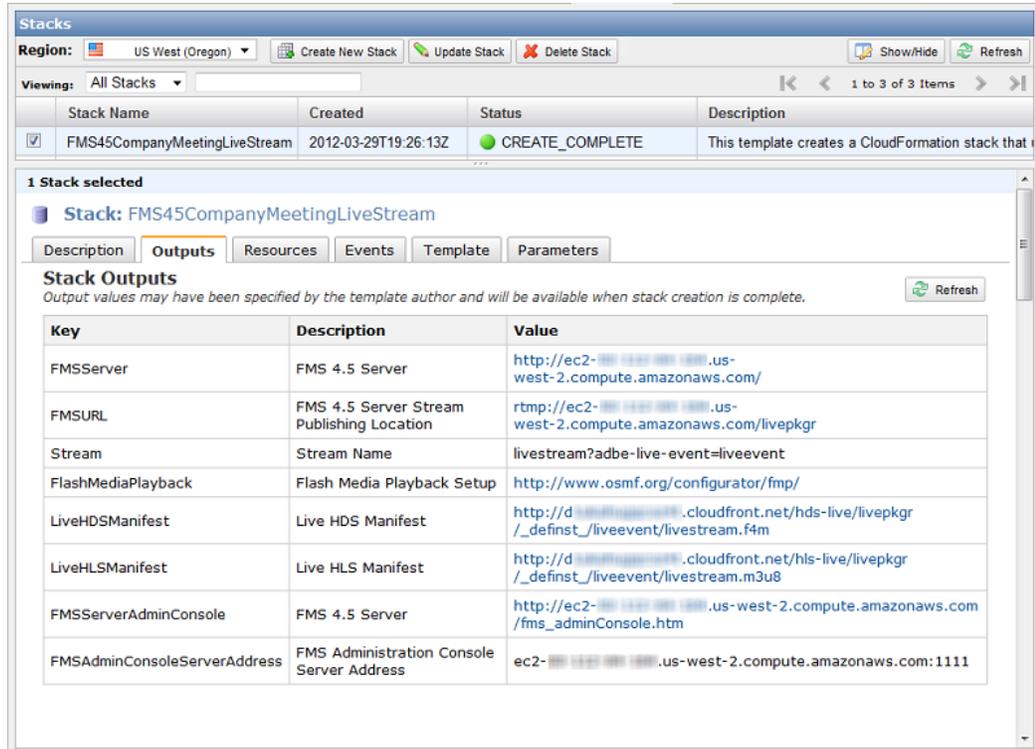
Next: Verifying that Adobe Flash Media Server Is Running (p. 147)

# Verifying that Adobe Flash Media Server Is Running

After AWS CloudFormation creates the stack, perform the following procedure to verify that Adobe Flash Media Server is running on the Amazon Amazon EC2 instance that you provisioned by using AWS CloudFormation.

**To verify that Adobe Flash Media Server is running**

1.  Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2.  In the top pane, select the stack that you created in Creating an AWS CloudFormation Stack for Live Streaming (p. 143).
3.  In the bottom pane, click the **Outputs** tab.
4.  Click on the value of the **FMSServer** key, for example,
    **http://ec2-00-11-22-33.us-west-1.compute.amazonaws.com**.
5.  The Adobe Flash Media Server page appears and begins streaming content, which shows that Adobe Flash Media Server is running.

    If streaming does not start, return to Overview of Live HTTP Streaming with Amazon Web Services (p. 139), and verify that the values you specified in the first five tasks are correct.

    If the values were all correct, but streaming still has not started, see How do I troubleshoot my Amazon EC2 instance if streaming doesn't start? (p. 159).

# Setting Up Adobe Flash Media Live Encoder to Publish a Live Stream

Adobe Flash Media Server on Amazon Web Services includes an application called livepkgr that packages published streams for delivery using HTTP Dynamic Streaming (HDS) and HTTP Live Streaming (HLS).

The following procedure shows how to set up Adobe Flash Media Live Encoder (FMLE) to publish your live stream to the livepkgr application on Adobe Flash Media Server 4.5.

> **Note**
>
> The Windows version of Flash Media Live Encoder doesn't support the AAC audio format. To add support for AAC, Adobe recommends that you purchase the MainConcept AAC encoder.

**To specify live-streaming settings in Flash Media Live Encoder**

1. Log on to the computer that you'll use to broadcast the live stream on the day of the event.
2. Open a web browser, and browse to the Adobe Flash Media Live Encoder page, http://www.adobe.com/products/flashmediaserver/flashmediaencoder.
3. Download and install the Flash Media Live Encoder.

   > **Note**
   >
   > Flash Media Live Encoder is free, but you need an Adobe account (also free) to download it.

4. Open the Flash Media Live Encoder `config.xml` file in a text editor. The default installation location is:

   - **Windows:** `C:\Program Files\Adobe\Flash Media Live Encoder 3.2`.
   - **Macintosh:** `Applications:Adobe:Flash Media Live Encoder 3.2`.

5. Set the value of the following `<enable>` element to `true`:

   ```
   <flashmedialiveencoder_config>
       ...
       <mbrconfig>
           ...
           <streamsynchronization>
               ...
               <!-- "true" to enable this feature, "false" to disable. -->
               <enable>true</enable>
   ```

6. Save the file.
7. Run Flash Media Live Encoder.
8. On the **Encoding Options** tab, in the Preset list, click **High Bandwidth (800 Kbps) — H.264**.
9. On the **Encoding Options** tab, under the **Audio** check box, in the **Format** list, click **AAC**.
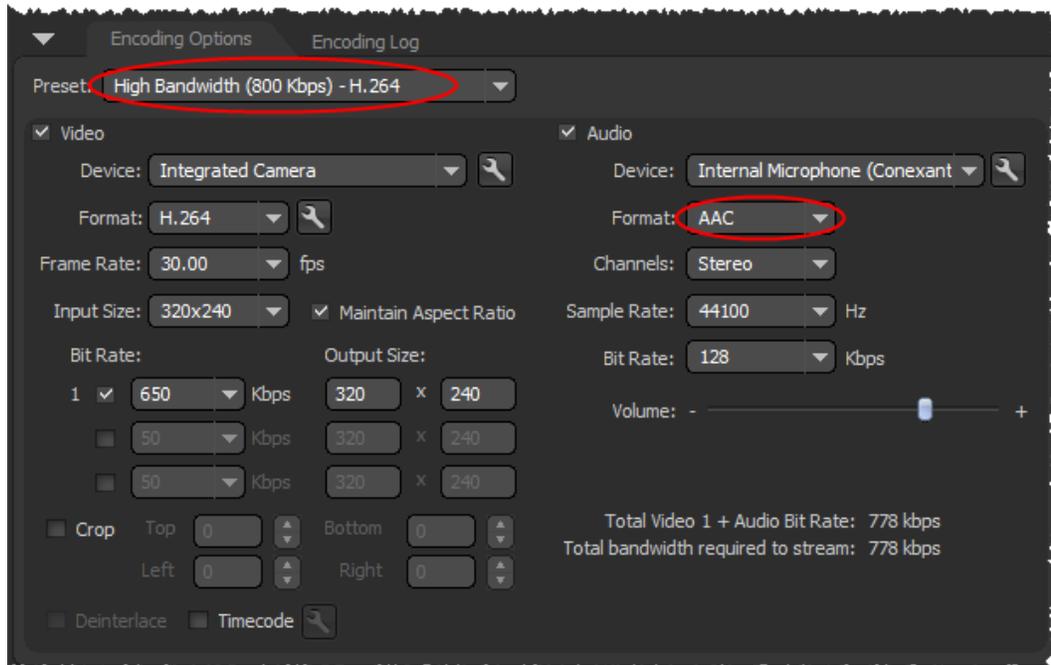
10. In the Video section of the Encoding Options tab, click the wrench icon on the right side of the Format list to open the Advanced Encoder Settings dialog box.
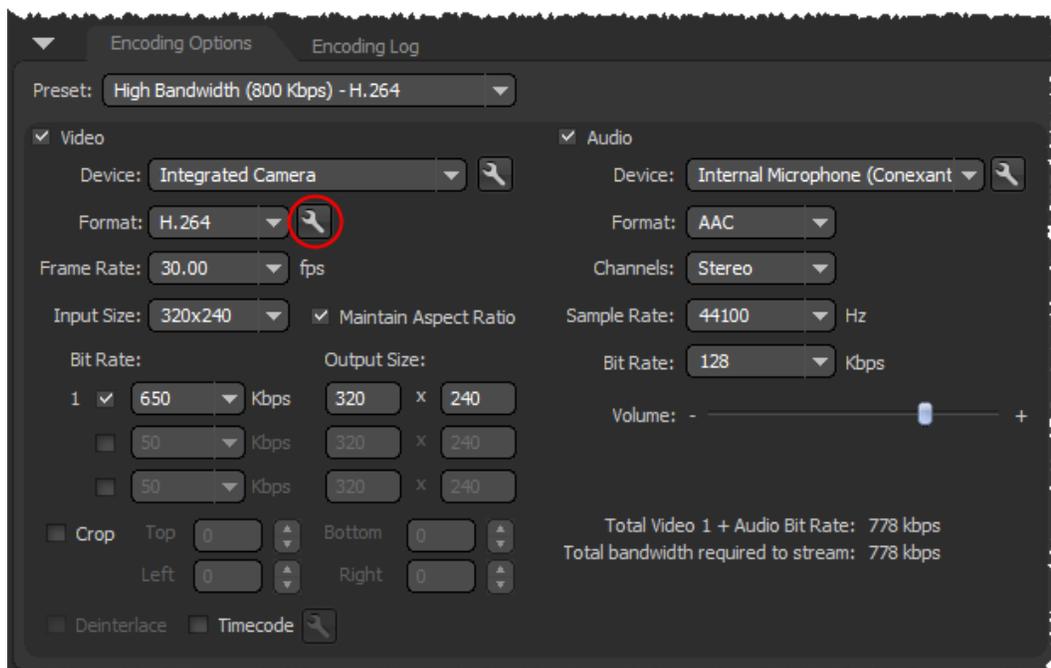


11. In the Advanced Encoder Settings dialog box, in the **Keyframe Frequency** list, click **4 Seconds**.

    You can also use a multiple of the value of the <FragmentDuration> element in the `applications/livepkgr/events/_definst_/liveevent/Event.xml` file. The default value of <FragmentDuration> is 4000 milliseconds (4 seconds).
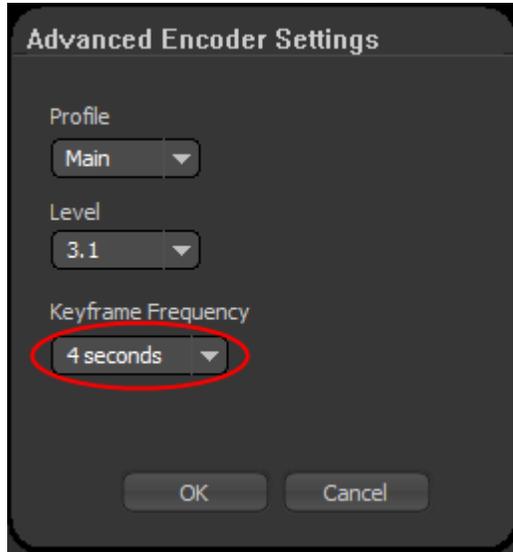
12. Click **OK** to save the setting and return to the main page. The value of the Preset list changes to `Custom`.

13. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.

14. Check the checkbox for the stack that you created for live streaming.

15. In the bottom pane of the AWS CloudFormation console, click the **Outputs** tab.

16. Copy the value of the **FMSURL** key, for example, *rtmp://ec2-00-11-22-33.us-west-1.compute.amazonaws.com/livepkgr*.

17. In Flash Media Live Encoder, in the Stream to Flash Media Server section, in the **FMS URL** field, paste the value of the FMSURL key that you copied from the AWS CloudFormation console.

18. In the AWS CloudFormation console, copy the value of the **Stream** key, for example, *livestream?adbe-live-event=liveevent*.

19. In Flash Media Live Encoder, in the **Stream** field, paste the value of the **Stream** key that you copied from the AWS CloudFormation console.

> **Note**
>
> If you anticipate having to stop and restart the live stream, enter the following value in the **Stream** field instead:
>
> ```
> livestream?adbe-live-event=liveevent&adbe-record-mode=record
> ```
>
> If you publish a live stream in record mode (`adbe-record-mode=record`), then stop the stream and restart it, Adobe Flash Media Server will delete the previous stream and start a new stream instead of appending to the previous stream when you restart. However, if you don't use record mode and you stop the live stream, you have to reconfigure live streaming before you can restart the stream.

**Amazon CloudFront Developer Guide**
**Embedding Flash Media Playback for an Amazon**
**CloudFront Live HTTP Stream in a Web Application**
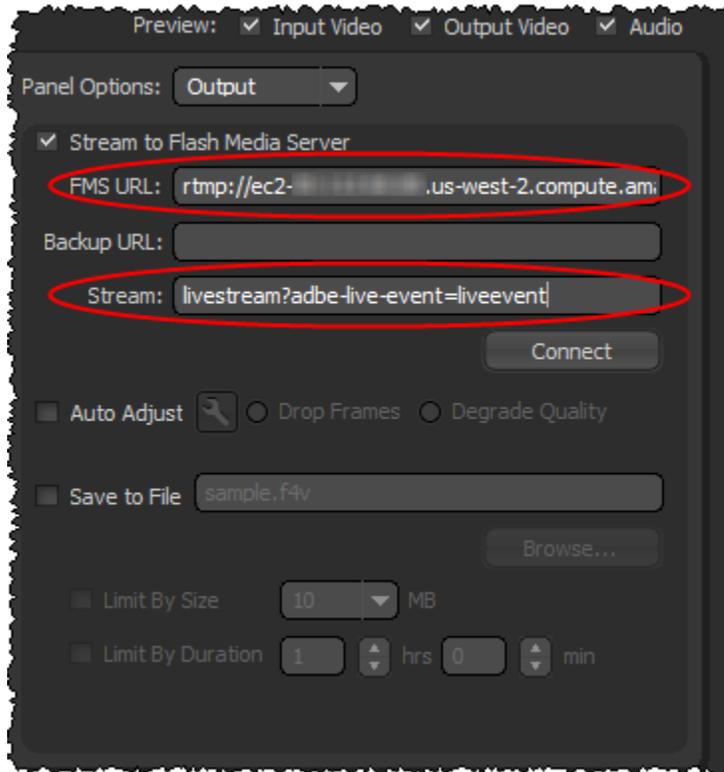
20. Uncheck the **Save to File** checkbox.
21. Click **Connect** to connect to your Flash Media Server instance.
22. Click **Start** to start encoding and publishing your live stream to the livepkgr application on your Adobe Flash Media Server instance.

Next: Embedding Flash Media Playback for an Amazon CloudFront Live HTTP Stream in a Web Application (p. 151)

# Embedding Flash Media Playback for an Amazon CloudFront Live HTTP Stream in a Web Application

Perform the applicable procedure to get the embed code that you will include in your web page for the live stream:

- To embed Flash Media playback for your HTTP stream via CloudFront (p. 151)
- To play your live HLS stream on an Apple device via CloudFront (p. 153)

**To embed Flash Media playback for your HTTP stream via CloudFront**

1. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. Check the checkbox for the stack for live streaming.
3. In the bottom pane of the AWS CloudFormation console, click the **Outputs** tab.
4. Copy the value of the **LiveHDSManifest** key, for example, *http://d123.cloudfront.net/hds-live/livepkgr/_definst_/liveevent/livestream.f4m*.
5. Click the value of the **FlashMediaPlayback** key to open the Flash Media Playback Setup webpage.

**Amazon CloudFront Developer Guide**
**Embedding Flash Media Playback for an Amazon**
**CloudFront Live HTTP Stream in a Web Application**

6.  On the Flash Media Playback Setup page, in the **Video Source** field, paste the value that you copied from the AWS Management Console in Step 4.

> **Note**
>
> Flash Media Playback requires Flash Player 10.1 to support HTTP Dynamic Streaming.



7.  For the **Are you using HTTP Streaming or Flash Access 2.0** radio buttons, click **Yes**.
8.  Delete the value in the **Poster frame file location** field, if any.
9.  Click **Preview** to update the value of the Preview Embed Code text box.

Preview Embed Code

```
<object width="600" height="409"> <param name="movie" value="http://fpdownload.adobe.com/strobe
/FlashMediaPlayback_101.swf"></param><param name="flashvars" value="src=http%3A%2F
%2Fd            .cloudfront.net%2Fhds-live%2Flivepkgr%2F_definst_%2Fliveevent%2Flivestream.f4m">
</param><param name="allowFullScreen" value="true"></param><param name="allowscriptaccess"
value="always"></param><embed src="http://fpdownload.adobe.com/strobe/FlashMediaPlayback_101.swf"
type="application/x-shockwave-flash" allowscriptaccess="always" allowfullscreen="true" width="600"
height="409" flashvars="src=http%3A%2F%2Fd            .cloudfront.net%2Fhds-
live%2Flivepkgr%2F_definst_%2Fliveevent%2Flivestream.f4m"></embed></object>
```

10. Play the video to ensure that you are satisfied with the current settings.
11. Change settings as desired. After you change options, click **Preview** to update the embed code.
12. To embed Flash Media Playback in a web page, copy the value of the **Preview Embed Code** text box, and paste it into the HTML code for your website.

> **Note**
>
> If you want users to view your live stream using a Flash-based player that is hosted on your own domain, see How do I update crossdomain.xml so my users can view the live stream using a Flash-based player that is hosted on my own domain? (p. 155).

**To play your live HLS stream on an Apple device via CloudFront**

1. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. Check the checkbox for the stack for live streaming.
3. In the bottom pane of the AWS CloudFormation console, click the **Outputs** tab.
4. Copy the value of the **LiveHLSManifest** key, for example,
   `http://d123.cloudfront.net/hls-live/livepkgr/_definst_/liveevent/livestream.m3u8.`

For information about where to use the URL to serve various iOS devices, QuickTime, and Safari, go to HTTP Live Streaming Overview in the iOS Developer Library.

For more information about publishing and playing live streams over HTTP, go to URLs for publishing and playing live streams over HTTP.

Next: Deleting an AWS CloudFormation Stack for Live Streaming (p. 153)

# Deleting an AWS CloudFormation Stack for Live Streaming

When your live event is over, delete the stack that you created for live streaming. This deletes the AWS resources that were created for your live-streaming event, and stops the AWS charges for the resources.

**To delete an AWS CloudFormation stack for live streaming**

1. Sign in to the AWS Management Console and open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. Check the checkbox for the stack, and click **Delete Stack**.
3. Click **Yes, Delete** to confirm.

4. To track the progress of the stack deletion, check the checkbox for the stack, and click the **Events** tab in the bottom frame.

5. If you do not plan to use live streaming again soon, you can cancel your subscription to Adobe Flash Media Server on Amazon EC2. To cancel the subscription, go to http://www.adobe.com/go/learn_fms_aws_order_en, and follow the on-screen prompts.

# Frequently Asked Questions

- How can I use Secure Shell (SSH) to connect to my Amazon EC2 instance that is running Adobe Flash Media Server 4.5? (p. 154)
- How do I update crossdomain.xml so my users can view the live stream using a Flash-based player that is hosted on my own domain? (p. 155)
- What is the price for live HTTP streaming using CloudFront and Adobe Flash Media Server 4.5? (p. 156)
- How can I create a CNAME alias for my Amazon EC2 instance or for my CloudFront distribution? (p. 156)
- How can I connect to the Flash Media Administration Console? (p. 156)
- Can I stream my live event both to Apple devices and to Flash Player–compatible devices? (p. 158)
- Does Flash Media Server 4.5 support HTML5? (p. 158)
- Does Flash Media Server have logging? (p. 158)
- How can I enable authentication on Flash Media Server? (p. 158)
- What are the default cache-control settings on HDS- and HLS-related files? (p. 158)
- What is the difference between HLS and HDS? (p. 159)
- How do I troubleshoot my Amazon EC2 instance if streaming doesn't start? (p. 159)
- Where can I find the documentation for live streaming using Adobe Flash Media Server 4.0? (p. 159)

# How can I use Secure Shell (SSH) to connect to my Amazon EC2 instance that is running Adobe Flash Media Server 4.5?

> **Note**
>
> By default, the SSH port for the Amazon EC2 instance (port 22) is disabled for security reasons. The following procedure explains how to enable the SSH port and how to use SSH to connect to your Amazon EC2 instance.

**To use SSH to connect to your Amazon EC2 instance that is running Adobe Flash Media Server 4.5**

1. Sign in to the AWS Management Console.

2. Authorize network access to your Amazon EC2 instance. For more information, see Authorize Network Access to Your Instances in the *Amazon Elastic Compute Cloud User Guide*.

3. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

4. In the Navigation pane, click **Instances**.

5. Right-click the correct instance, and click **Connect** to view instructions on how to use SSH to connect to your Amazon EC2 instance:
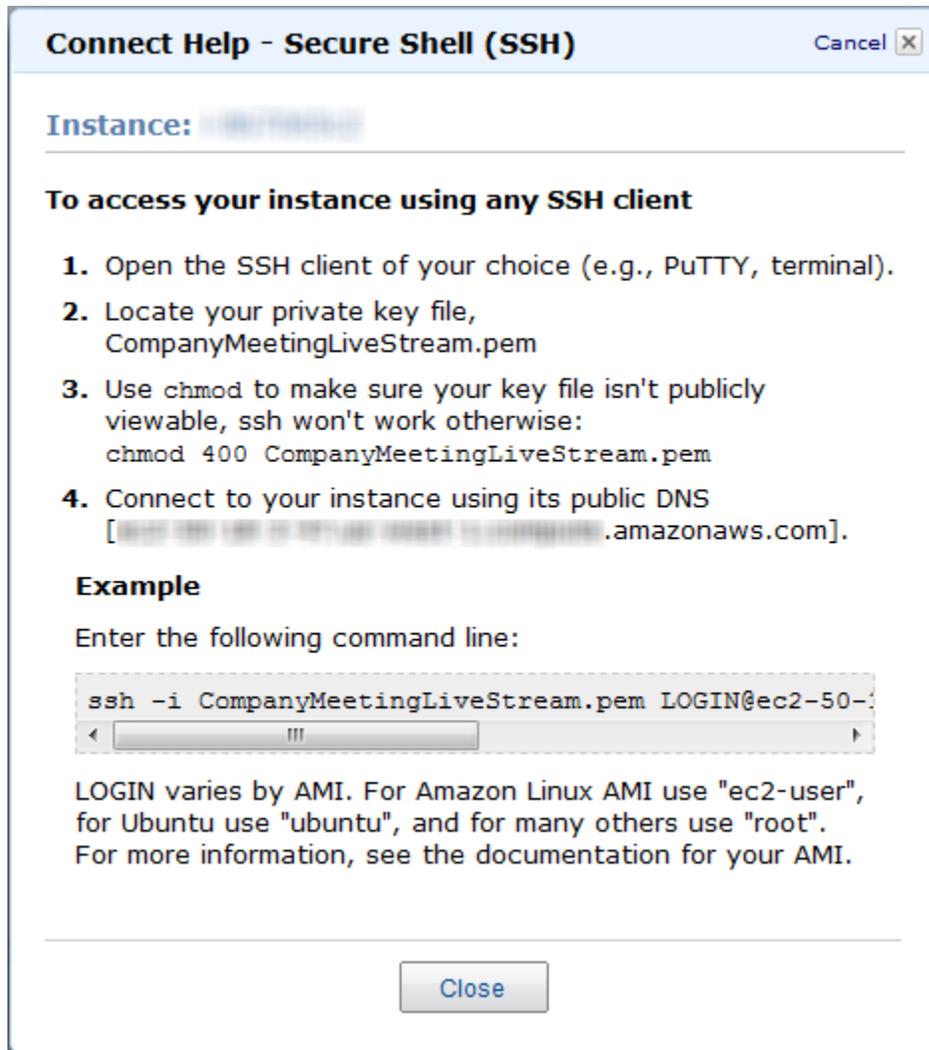
# How do I update crossdomain.xml so my users can view the live stream using a Flash-based player that is hosted on my own domain?

You can change permissions in `crossdomain.xml` either before or after you create the AWS CloudFormation stack:

- If you have not created your AWS CloudFormation stack, download the AWS CloudFormation template for Live Streaming using Amazon CloudFront and Adobe Flash Media Server 4.5 at https://cloudfront-live.s3.amazonaws.com/live-http-streaming-fms-4-5-1-using-cloudfront.txt. In the template, edit the `UserData` section, which contains the `crossdomain.xml` settings, and save the updated template on your local computer. Then create your AWS CloudFormation stack using the updated template.
- If you have already created your AWS CloudFormation stack, log into Adobe Flash Media Server running on your Amazon EC2 instance, and change permissions in the cross-domain policy file, `/mnt/webroot/crossdomain.xml`.

For more information about editing the `crossdomain.xml` file, go to Adobe Cross Domain Policy File Specification.

# What is the price for live HTTP streaming using CloudFront and Adobe Flash Media Server 4.5?

In addition to the $5.00 monthly subscription fee for Adobe Flash Media Server on Amazon EC2, you pay only for the AWS resources you consume:

*   For pricing information about Adobe Flash Media Server running on Amazon EC2, see the **Pricing** tab on the Adobe Flash Media Server on Amazon Web Services web page at http://www.adobe.com/ap/products/flashmediaserver/amazonwebservices/.
*   For pricing information about CloudFront, see http://aws.amazon.com/cloudfront/pricing.

There is no charge for using AWS CloudFormation.

# How can I create a CNAME alias for my Amazon EC2 instance or for my CloudFront distribution?

Your Amazon EC2 instance running Adobe Flash Media Server 4.5 comes with an internal and an external DNS name. Amazon EC2 does not provide access to modify these DNS settings. If you want to map an existing domain name to your Amazon EC2 instance running Flash Media Server, use a DNS service provider such as Amazon Route 53. When using your own domain name, we recommend that you map to the instance's external DNS name using a CNAME, not by using an A record that points to the instance's IP address.

To map your own domain name to your CloudFront distribution, see Using CNAMEs (p. 49).

# How can I connect to the Flash Media Administration Console?

**To connect to the Flash Media Administration Console**

1.  Sign in to the AWS Management Console and open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2.  Select the stack for live streaming.
3.  In the bottom pane of the AWS CloudFormation console, click the **Outputs** tab.
4.  Click the value of the **FMSServerAdminConsole** key, for example, **http://ec2-00-11-22-33.us-west-1.compute.amazonaws.com/fms_adminConsole.htm**.
5.  In the AWS CloudFormation console, copy the value of the **FMSAdminConsoleServerAddress** key to the clipboard.
6.  In the **Server Address** field, paste the value that you copied in the previous step.
7.  In the **Username** and **Password** fields, enter the values that you specified in Creating an AWS CloudFormation Stack for Live Streaming (p. 143).
8.  Click **Login**.

For information about using the Flash Media Server 4.5 Administration Console, refer to the Adobe documentation.

> **Note**
>
> Adobe recommends that you block all external access to port 1111 so that access to the
> Administration Console is restricted only to clients that are within your firewall. As an alternative,
> you can restrict access to the server by using domain-based restrictions. For more information,
> go to the Limit access to Flash Media Administration Server in the Adobe documentation.

**To disable or restrict access to port 1111 on your Flash Media Server**

1.  Get the name of the Amazon EC2 security group that is associated with your Amazon EC2 instance:

    a.  Sign in to the AWS Management Console and open the AWS CloudFormation console at
        https://console.aws.amazon.com/cloudformation/.
    b.  In the Region list, click the name of the region in which you created your Amazon EC2 instance.
    c.  Click the row for your AWS CloudFormation stack.
    d.  In the bottom pane, click the **Resources** tab.
    e.  In the left column of the **Stack Resources** table, find the row for which the value is
        `FMSOriginServerSecurityGroup.`
    f.  For that row, write down the value of the **Physical ID** column.


2.  Display the Amazon EC2 console.
3.  In the **Navigation** pane, click **Security Groups**.
4.  In the **Security Groups** pane, click the row for which the value of the **Name** column matches the
    value that you got in Step 1f.
5.  In the bottom pane, click the **Inbound** tab.
6.  To completely disable access to the Flash Media Administration console:

    a.  In the **TCP Port (Service)** column, find **1111**.
    b.  In the **Action** column for that row, click **Delete**.
    c.  Click **Apply Rule Changes**.


    To restrict access to selected IP addresses:

    a.  In the **TCP Port (Service)** column, find **1111**.
    b.  In the **Action** column for that row, click **Delete**.
    c.  In the **Create a new rule** list, do not change the default value, **Custom TCP rule**.
    d.  In the **Port range** field, enter **1111**.
    e.  In the **Source** field, enter an IP address or range, or enter the name of another security group.
        For more information, click **Help**.
    f.  Click **Add Rule**.
    g.  To create additional rules, repeat Steps c through f.
    h.  Click **Apply Rule Changes**.

# Can I stream my live event both to Apple devices and to Flash Player–compatible devices?

Yes, Flash Media Server 4.5 enables the delivery of live streams to both Flash-based and iOS devices at the same time. You can stream to the Safari browser using an HTML5 player or an Objective C ("native") application. You can also use Adobe AIR for iOS to develop a rich video experience on iOS.

# Does Flash Media Server 4.5 support HTML5?

Yes. Flash Media Server can deliver content to HTML5 on Apple iOS devices using the HLS streaming format. For other browsers supporting HTML5, you can use Flash Media Server to deliver progressively.

# Does Flash Media Server have logging?

Yes. W3C-compliant ASCII logs, a real-time usage monitor, and a complete API for server and stream events help to ensure that you have all the tools you need to track and generate reports on your audience's content use. For more information about monitoring and managing log files in Flash Media Server 4.5, go to Monitoring and Managing Log Files in the Adobe documentation.

# How can I enable authentication on Flash Media Server?

You can download the Authentication add-in and install it on your Flash Media Server instance. You can also restrict access to RTMP port 1935 (for both TCP and UDP) in the security group created by AWS CloudFormation for your Flash Media Server Amazon EC2 instance. Just create new TCP and UDP rules for port 1935 and then delete the existing TCP and UDP rules for port 1935, which allow access to all IP addresses.

For a quick overview of how to add a rule to a security group, see How can I connect to the Flash Media Administration Console? (p. 156). For more information about Amazon EC2 security groups, go to the Using Security Groups in the *Amazon Elastic Compute Cloud User Guide.*

# What are the default cache-control settings on HDS- and HLS-related files?

The default cache control headers on HDS- and HLS-related files are set to the following values:

| File Type | `Cache-Control` Setting (Seconds) |
| --- | --- |
| .bootstrap | 2 |
| .f4f | none (the CloudFront default value is 86400 seconds, or one day) |
| .f4m | 2 |
| .m3u8 | 2 |
| .ts | none (the CloudFront default value is 86400 seconds, or one day) |

The CloudFront edge cache servers honor these cache control headers. You can change the default settings by changing the values of the `HttpStreamingF4MMaxAge`, `HttpStreamingBootstrapMaxAge`, and `HttpStreamingFragMaxAge` parameters on the server. For more information, go to HTTP streaming configuration file reference on the Adobe website.

# What is the difference between HLS and HDS?

HLS is a file container format optimized for Apple devices. The container supports H.264/AAC-encoded video and audio, and is based on MPEG-2 transport stream (TS). All video delivered to iOS (including AIR for IOS) must use this format.

HDS is a file container format optimized for applications that run in Flash Player. The container also supports H.264/AAC-encoded video and audio and is based on MPEG-4 TS. HDS is not supported on AIR for iOS.

# How do I troubleshoot my Amazon EC2 instance if streaming doesn't start?

If you performed the procedure To verify that Adobe Flash Media Server is running (p. 147) and streaming still hasn't started, perform the following procedure to confirm that the Amazon EC2 instance is functioning correctly.

**To troubleshoot your Amazon EC2 instance running Adobe Flash Media Server 4.5**

1.  In the AWS CloudFormation console, in the top pane, select the stack.
2.  In the bottom pane, click the **Resources** tab.
3.  For the **FMSOriginServer** row, write down the value of the **Physical ID** column.
4.  Go to the Amazon EC2 console.
5.  In the **Navigation** pane, select the region in which you created the AWS CloudFormation stack.
6.  In the **Navigation** pane, click **Instances**.
7.  In the **Instance** column, find the value that you wrote down in Step c.
8.  Select the corresponding row.
9.  In the bottom pane, review the information on the **Status Checks** tab, and take the recommended actions.
10. Return to the procedure To verify that Adobe Flash Media Server is running (p. 147), and repeat Steps 2 through 5.

# Where can I find the documentation for live streaming using Adobe Flash Media Server 4.0?

For the documentation for the original release of live streaming, using Adobe Flash Media Server 4.0, see Live Streaming Using CloudFront and Adobe Flash Media Server.

# Additional Documentation

## Adobe Documentation

- Using Adobe Flash Media Server on Amazon Web Services,
  http://www.adobe.com/products/flashmediaserver/amazonwebservices/
- Adobe Cross Domain Policy File Specification,
  http://learn.adobe.com/wiki/download/attachments/64389123/CrossDomain_PolicyFile_Specification.pdf
- Adobe Flash media Live Encoder:
  - http://www.adobe.com/products/flashmediaserver/flashmediaencoder
  - http://www.adobe.com/products/flash-media-encoder/faq.html

- Video: Encoding and Transcoding Recommendations for HTTP Dynamic Streaming on the Adobe Flash Platform, http://download.macromedia.com/flashmediaserver/http_encoding_recommendations.pdf
- *Adobe Flash Media Server 4.5 Technical Overview*, http://help.adobe.com/en_US/flashmediaserver/techoverview/flashmediaserver_4.5_tech_overview.pdf.

## Amazon Web Services Documentation

- Amazon Elastic Compute Cloud, http://aws.amazon.com/documentation/ec2
- CloudFront, http://aws.amazon.com/documentation/cloudfront
- AWS CloudFormation, http://aws.amazon.com/documentation/cloudformation

# Live Smooth Streaming Using Amazon CloudFront and IIS Media Services 4.1

**Topics**

## Overview of Live Smooth Streaming with Amazon Web Services

Smooth Streaming is the Microsoft implementation of adaptive streaming technology, which is a form of web-based media content delivery that uses standard HTTP. An extension of IIS Media Services, Smooth Streaming enables adaptive streaming of live events to Smooth Streaming clients such as Microsoft Silverlight. When you configure Smooth Streaming to use CloudFront, you benefit from the scale of CloudFront's global HTTP network and from latency-based routing of viewers to edge nodes on the network. To learn more about CloudFront, go to the CloudFront product page.

Smooth Streaming content is delivered to clients as a series of MPEG-4 (MP4) fragments that can be cached at the CloudFront edge servers. Smooth Streaming–compatible clients use special heuristics to dynamically monitor current network and local PC conditions, and seamlessly switch the video quality of the Smooth Streaming presentation that the clients receive. As clients play the fragments, network conditions may change (for example, bandwidth may decrease) or video processing may be affected by other applications that are running on the client. Clients can immediately request that the next fragment come from a stream that is encoded at a different bit rate to accommodate the changing conditions. This enables clients to play the media without stuttering, buffering, or freezing. As a result, users experience the highest-quality playback available without interruptions in the stream.

To encode a live broadcast to Smooth Streaming format, you use Microsoft Expression Encoder 4 Pro. To serve the encoded Smooth Stream, you can then use an Amazon EC2 Amazon Machine Image (AMI) that is running Windows IIS Media Services. CloudFront caches the live video and audio content, and viewers connect to the CloudFront edge servers to play the stream using a Smooth Streaming-compatible client such as Microsoft Silverlight. This tutorial walks you through the entire setup process.

To set up Live Smooth Streaming with Amazon Web Services (AWS), review the system requirements for IIS Smooth Streaming in the *Smooth Streaming Deployment Guide*. Then perform the procedures in the following sections:

1. Creating an Amazon Web Services Account (p. 162)
2. Creating an Amazon EC2 Key Pair (p. 162)
3. Creating an AWS CloudFormation Stack for Live Smooth Streaming (p. 163)
4. Verifying that Your Amazon EC2 Windows Server Instance Is Running (p. 166)

For frequently asked questions, see Frequently Asked Questions (p. 175).

For links to additional Microsoft and AWS documentation, see Additional Documentation (p. 177).

# Creating an Amazon Web Services Account

If you already have an AWS account, skip to Creating an Amazon EC2 Key Pair (p. 162). If you don't already have an AWS account, use the following procedure to create one.

> **Note**
>
> When you create an account, AWS automatically signs up the account for all services. You are charged only for the services you use.

**To create an AWS account**

1.  Go to http://aws.amazon.com, and click **Create an AWS Account**.
2.  Follow the on-screen instructions.

    Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.

# Creating an Amazon EC2 Key Pair

If you already have an Amazon EC2 key pair in the Amazon EC2 region in which you want to configure Live Smooth Streaming, skip to Creating an AWS CloudFormation Stack for Live Smooth Streaming (p. 163). If you don't have a key pair in that region, perform the following procedure.

A key pair is a security credential similar to a password. You specify a key pair when you create an AWS CloudFormation stack for live streaming, later in this process. After live streaming is configured, you use the key pair to retrieve the password for your Amazon EC2 Windows Server instance.

**To create an Amazon EC2 key pair**

1.  Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2.  In the Navigation pane, in the Region list, click the region in which you want to create the key pair.
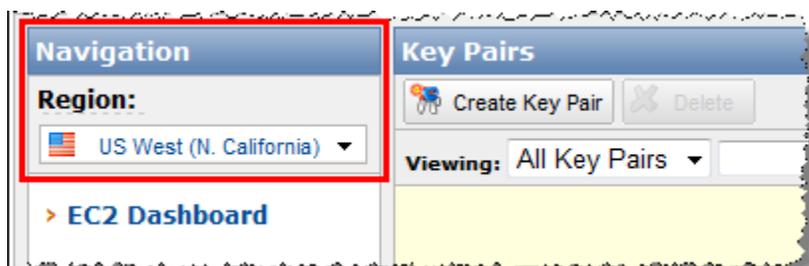
You must create the key pair in the same region where you will create your AWS CloudFormation stack for live streaming later in this process. We recommend that you create the key pair and the stack for live streaming in the region that is closest to the location of your live event.

3. In the Navigation pane, click **Key Pairs**.
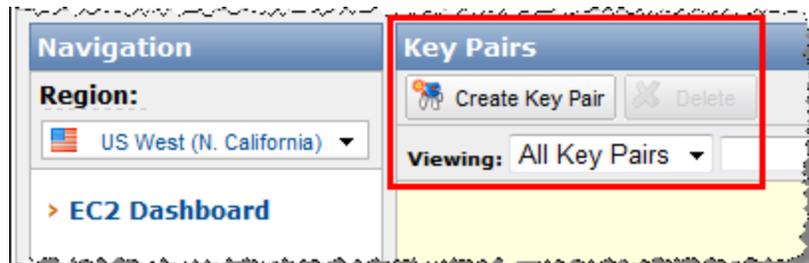
4. In the Key Pairs pane, click **Create Key Pair**.



5. In the Create Key Pair dialog box, enter a name for the key pair, and make note of the name. You'll enter this value when you create an AWS CloudFormation live-streaming stack, later in the process of setting up live streaming.

6. Click **Create**, and the Opening <key_pair_name>.pem dialog box appears.

7. Save the .pem file to a safe place on your computer.

8. Click **Close** to close the Create Key Pair dialog box.

# Creating an AWS CloudFormation Stack for Live Smooth Streaming

The following procedure uses an AWS CloudFormation template to create a stack that launches the AWS resources required for Live Smooth Streaming.

**To create an AWS CloudFormation stack for live streaming**

1. In the following list, click the Amazon EC2 Region where you want to create the stack. The Create Stack wizard starts, and a region-specific value is automatically entered in the **Provide a Template URL** field.

   US East (Virginia)

   US West (Oregon)

   US West (Northern California)

   EU (Ireland)

   Asia Pacific (Singapore)

   Asia Pacific (Tokyo)

   South America (Sao Paulo)

2. If you are not already signed in to the AWS Management Console, sign in when prompted.

3. *Optional:* In the Create Stack wizard, change the value of the **Stack Name** field. The stack name must not contain spaces, and it must be unique within your AWS account.

4. Do not change the **Stack Template Source** option or the value of **Provide a Template URL**.

5. *Optional:* To configure SNS notification, to specify how long you're willing to wait for the stack to be created, and to choose whether to roll back changes if stack creation fails, check the **Show Advanced Options** checkbox, and specify the applicable values.

6. Click **Continue**.



7. On the Specify Parameters page, in the **KeyPair** field, enter the name of an Amazon EC2 key pair in the region in which you want to create the stack for live streaming. The key pair must be associated

with the account that you're currently logged on with. If you created a key pair when you performed the procedure in Creating an Amazon EC2 Key Pair (p. 162), enter the name of that key pair.

8. In the **InstanceType** field, enter an instance type, and click **Continue**. The default value is *m1.xlarge*.

   The instance type determines the pricing for your Amazon EC2 instance that is running Windows Server. For more information about Amazon EC2 instance types for Windows, including pricing information, go to Amazon EC2 Running Microsoft Windows Server & SQL Server.

9. Review the settings for the stack. When you're satisfied with the settings, click **Create Stack**.

   Your stack may take several minutes to create. To track the progress of the stack creation, select the stack, and click the **Events** tab in the bottom frame. If AWS CloudFormation cannot create the stack, the Events tab lists error messages.

   When your stack is ready, in the top frame, the status for the stack changes to **CREATE_COMPLETE**.



   When your stack is created, click the **Outputs** tab, which displays the stack creation outputs. You will use these values when you set up Microsoft Expression Encoder later in the process.
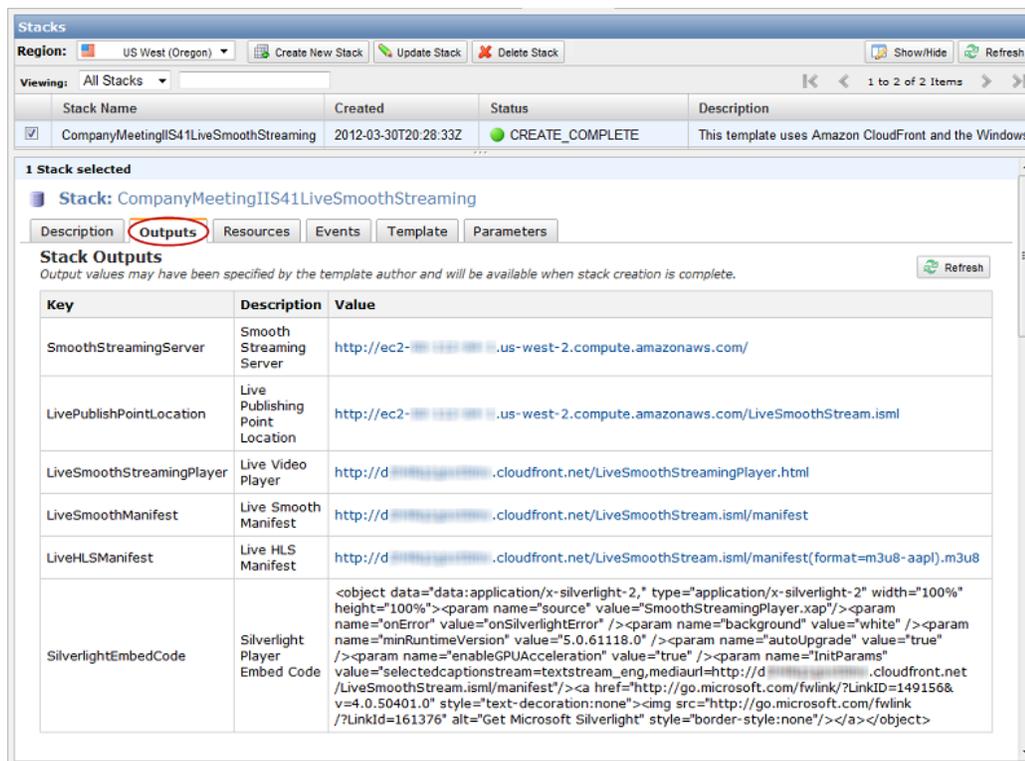
# Verifying that Your Amazon EC2 Windows Server Instance Is Running

After AWS CloudFormation creates the stack, perform the following procedure to verify that your Windows IIS Media Services webserver is running on the Amazon EC2 instance that you provisioned via AWS CloudFormation.

**To verify that your Windows Server is running**

1. Sign in to the AWS Management Console and open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. In the top pane, select the stack that you created in Creating an AWS CloudFormation Stack for Live Smooth Streaming (p. 163).
3. In the bottom pane, click the **Outputs** tab.
4. Click on the value of the **SmoothStreamingServer** key, for example, **http://ec2-00-11-22-33.us-west-1.compute.amazonaws.com**.

   The Windows IIS Server banner screen appears, indicating that your Windows Server is running.

# Getting Your Windows Password

To connect to your Amazon EC2 instance running Windows Server 2008 R2 and IIS Media Services, use the following procedure to retrieve the initial password for the Windows Server Administrator account. You only need to retrieve the password once for your Amazon EC2 instance. When you are finished with this procedure, you'll be able to work with your Amazon EC2 instance as you would any Windows Server computer.

For more information about connecting to an Amazon EC2 instance running Windows, go to Getting Started Guide AWS Computing Basics for Windows.

> ⚠️ **Important**
>
> Amazon EC2 can take as long as 30 minutes to retrieve your password from Windows Server.

**To get the Windows password for your Amazon EC2 instance**

1. Confirm that you can access the Amazon EC2 private key file (the .pem file) that you created in Creating an Amazon EC2 Key Pair (p. 162).
2. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
3. In the **Navigation** pane, in the **Region** list, click the region in which you created the Amazon EC2 instance for Live Smooth Streaming.
4. In the **Navigation** pane, click **Instances**.
5. In the My Instances pane, right-click the instance for which the value of the **Name** column is **LiveSmoothStreaming**, and click **Get Windows Password**.
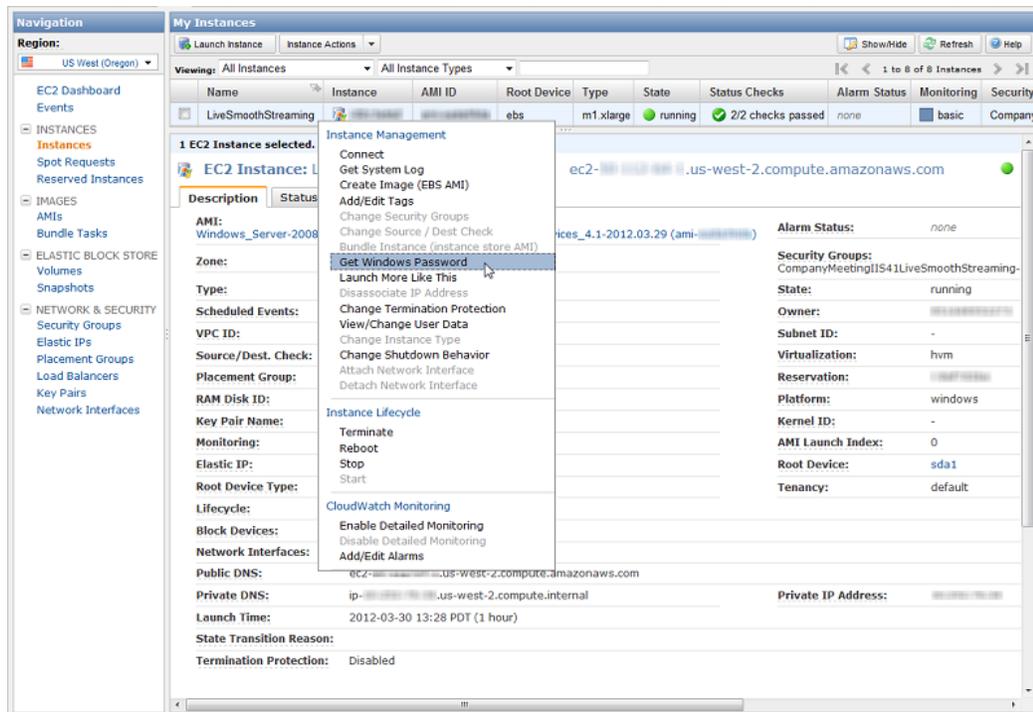


6. On the Retrieve Default Windows Administrator Password page, click **Browse**, and browse to the location on your computer where you saved the .pem file.
7. Select the .pem file, and the contents of the file appear in the window.

8.  Click **Decrypt Password**.
9.  Write down the password. You'll need it to connect to the Amazon EC2 instance.
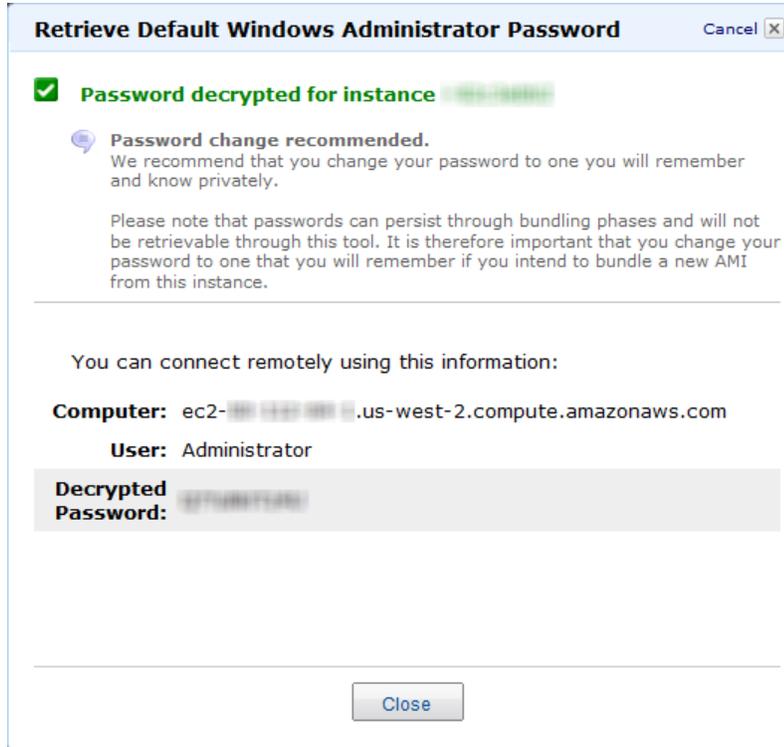
10. *Optional but recommended:* Log into the Windows Server instance that you just launched, and change the password for the default Windows Server account. The username is Administrator.

   You may also want to create another user account and add it to the Administrators group. Another administrator account is a safeguard in case you forget your administrator password or have a problem with the Administrator account.

> 📑 **Note**
>
> For information about how to update the Amazon EC2 Security Group settings for your Windows server so you can access the server using port 3389, see How can I enable access to the Windows server? (p. 176). For information about how to log on to the instance using the Administrator account, see How can I securely connect to my Amazon EC2 instance running Windows IIS Media Services? (p. 176).

# Encoding Your Live Stream

Use the procedure in this section to create a Live Broadcasting Project using Microsoft Expression Encoder 4 Pro SP2 and to publish your live stream to the Live Smooth Streaming publishing point on your Amazon EC2 instance running Windows Server and Windows IIS Media Services.

To learn more about live broadcasting using Microsoft Expression Encoder, go to Creating a Live Broadcasting Project on the Microsoft Expression website.

> 📝 **Note**
>
> Microsoft Expression Encoder 4 Pro with Service Pack 2 is not a free download. For more information about features and pricing, go to the Expression Encoder 4 Pro page on the Microsoft website.

You can also use a third-party encoding tool to encode your video for Live Smooth Streaming. For a list of Microsoft partners that provide encoding software, see the Partners tab on the IIS Media Services page on the Microsoft website.

### To encode a live broadcast

1.  Log into the Amazon EC2 instance that you created for live streaming.
2.  On the Windows Start menu, click **All Programs > Microsoft Expression > Microsoft Expression Encoder 4**.
3.  In the **Load a new project** dialog box, click **Live Broadcasting Project**.



4.  Click **Add a Live Source** to use for your live broadcast.
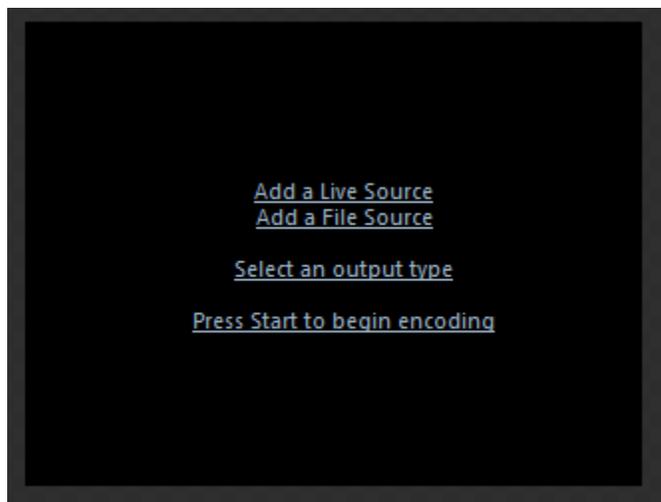
> **Note**
>
> You can connect multiple camera devices, such as USB webcams or FireWire (IEEE 1394) digital video cameras. Although you can connect multiple live sources, you can stream only one at a time. For more information about setting up sources for live broadcasts, go to Set Live Sources on the Microsoft Expression website.

5.  On the **Presets** tab, choose the encoding preset that supports the bit rates and encoding requirements for your Live Smooth Streaming scenario. Choose an option that has **IIS Smooth Streaming** in the name.
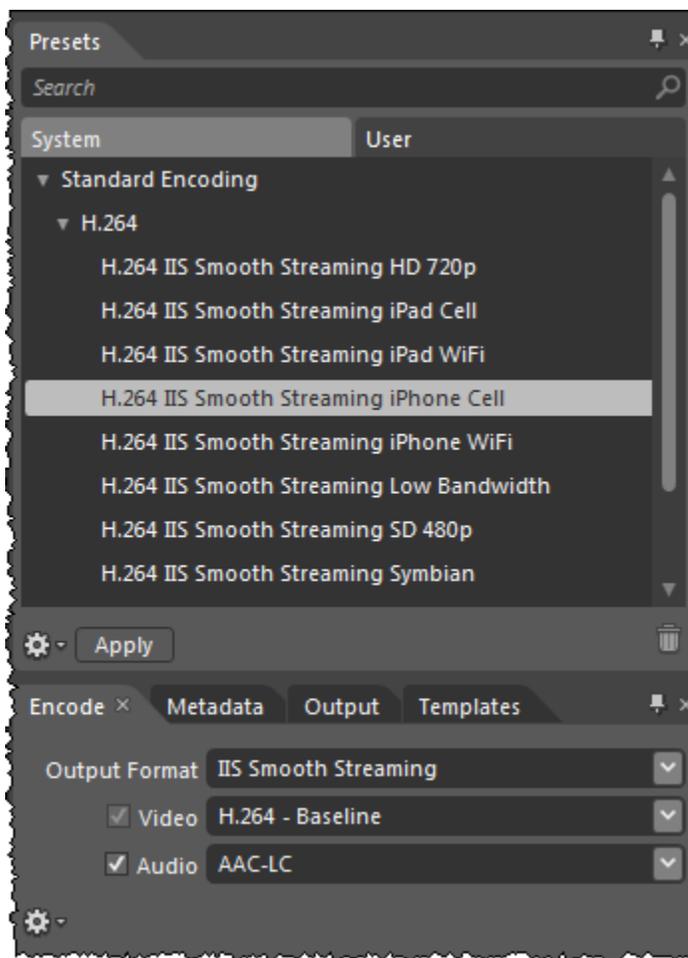
When you click **Apply**, the **Output Format**, **Video**, and **Audio** settings on the **Encode** tab are automatically updated with the values in the encoding preset that you selected.

For more information about a preset, for example, the number of streams in the output and the codecs used, hover your mouse pointer over a preset name.

📒 **Note**

> Alternatively, you can specify custom settings on the **Encode** tab. For more information, go to the following topics on the Microsoft Expression website:

- Set Output formats
- Video settings
- Audio settings

6. In Microsoft Expression Encoder, click the **Output** tab.
7. On the **Output** tab, check the **Publishing Point** check box.
8. Sign in to the AWS Management Console and open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
9. In the bottom pane of the AWS CloudFormation console, click the **Outputs** tab.

10. Copy the value of the **LivePublishPointLocation** key, for example,
**http://ec2-00-11-22-33.us-west-1.compute.amazonaws.com/LiveSmoothStream.isml**.

11. In Microsoft Expression Encoder, paste the URL that you copied in the previous step into the **Location**
field.
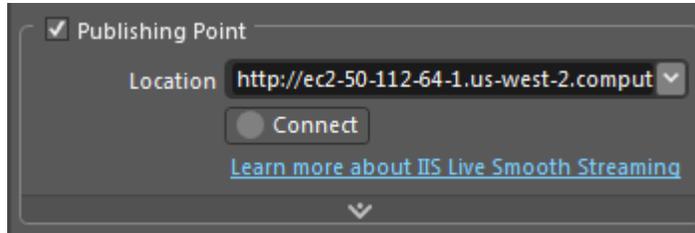


12. Click **Connect** to initiate a connection to the publishing point on your Windows server.

13. When you're prompted for your Publishing Point Administrator Password, enter the following values:

- **User Name:** Administrator
- **Password:** the Windows Server password that you retrieved in Getting Your Windows
Password (p. 167).

Then click **OK**.

> 📔 **Note**
>
> Windows authentication is configured for the default web site on your Windows server so
> you can connect to the live publishing point on the server from Microsoft Expression Encoder
> 4 Pro SP2. To learn more about Windows authentication, go to the IIS website. To learn
> about the authentication mechanisms available in IIS 7, go to the Microsoft website.

14. When a connection is successfully established, the publishing point state changes to **Starting**. In
addition, a **Restart** button appears next to the **Connect** button, below the **Location** field in the
Publishing Point section.
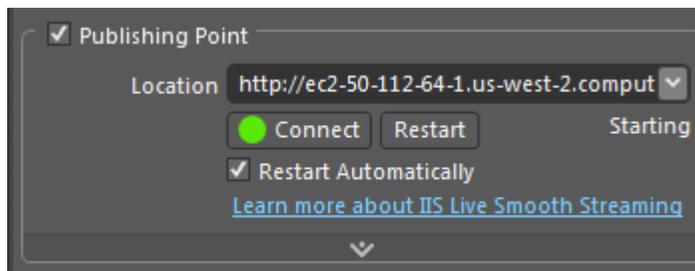


> 📔 **Note**
>
> The Starting state means that the publishing point is ready to receive live streams. When
> the live source connects to the publishing point and begins pushing content to it, the Starting
> state changes to Started, meaning that the publishing point is receiving the live streams.

> 📝 **Note**
>
> Microsoft Expression Encoder 4 Pro with SP2 utilizes the REST APIs that Windows IIS Media Services 4.1 includes to help you manage live publishing points on the Windows server. For more information, go to the IIS blog.

15. Click **Start** to begin encoding and publishing your live broadcast to the publishing point on your Amazon EC2 instance that is running Windows Server and IIS Media Services.
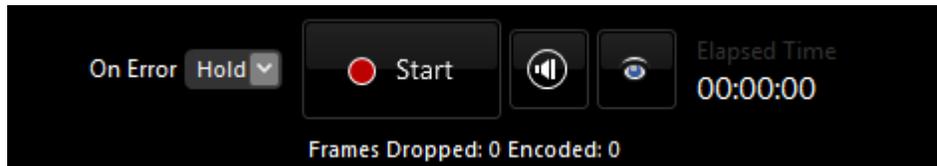


As the broadcast runs, you can monitor statistics and connections data in the corresponding panels. For more information about how to monitor this data, see the following topics on the Microsoft Expression website:

- Using the Statistics panel
- Using the Connections panel

# Viewing Your Live Smooth Stream

Perform the following procedure to view your live smooth stream using CloudFront. You can also embed the Microsoft Silverlight player code in your own web page.

1. Sign in to the AWS Management Console and open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. Select the stack for live streaming.
3. In the bottom pane of the AWS CloudFormation console, click the **Outputs** tab.
4. Click the value of the **LiveSmoothStreamingPlayer** key, for example, **http://d123.cloudfront.net/LiveSmoothStreamingPlayer.html**.
5. To embed the Silverlight player code into your web page, on the **Outputs** tab, copy the value of the **SilverlightEmbedCode** key.

> 📝 **Note**
>
> Microsoft recommends that viewers have the latest version of Microsoft Silverlight installed for the best playback experience.

6. To view your live stream on an Apple device such as an iPad or an iPhone, display the AWS CloudFormation console from a compatible Apple device, and click the value of the **LiveHLSManifest** key. The manifest URL looks like **http://d123.cloudfront.net/LiveSmoothStream.isml/manifest(format=m3u8-aapl).m3u8**.

   For information about where to use the URL to serve various iOS devices, QuickTime, and Safari, go to HTTP Live Streaming Overview in the iOS Developer Library.

# Deleting Your AWS CloudFormation Live Smooth Streaming Stack

When your live event is over, delete the stack that you created for Live Smooth Streaming. This deletes the AWS resources that were created for your live event, and stops the AWS charges for those resources.

**To delete an AWS CloudFormation stack for live streaming**

1. Sign in to the AWS Management Console and open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. Check the checkbox for the stack, and click **Delete Stack**.
3. Click **Yes, Delete** to confirm.
4. To track the progress of the stack deletion, check the checkbox for the stack, and click the **Events** tab in the bottom frame.

# Frequently Asked Questions

- What is the price for Live Smooth Streaming using CloudFront? (p. 175)
- Can I deliver my live streaming video to both Smooth Streaming clients and Apple devices? (p. 175)
- How can I set-up a CNAME alias for my Amazon EC2 instance or my CloudFront distribution? (p. 176)
- How can I enable access to the Windows server? (p. 176)
- How can I securely connect to my Amazon EC2 instance running Windows IIS Media Services? (p. 176)
- How can I restrict access to my Live Smooth Streaming content from another domain? (p. 177)

## What is the price for Live Smooth Streaming using CloudFront?

To Smooth Stream your live event, you pay only for the AWS resources you consume:

- For pricing information about Amazon EC2 instances running Windows Server, see **Pricing** on the Amazon EC2 Running Microsoft Windows Server & SQL Server page.
- For pricing information about CloudFront, see Amazon CloudFront Pricing.

There is no charge for using AWS CloudFormation.

## Can I deliver my live streaming video to both Smooth Streaming clients and Apple devices?

Yes. You can use Microsoft Expression Encoder 4 Pro to encode your live video for both Smooth Streaming clients (for example, Microsoft Silverlight) and Apple devices (for example, iPad and iPhone). After your AWS CloudFormation stack is launched, you will find the manifest file URLs both for Live Smooth Streaming (.isml) and for Apple HLS (.m3u8) on the **Outputs** tab of your AWS CloudFormation template.

# How can I set-up a CNAME alias for my Amazon EC2 instance or my CloudFront distribution?

Your Amazon EC2 Windows Server instance comes with an internal and an external DNS name. Amazon EC2 does not provide access to modify these DNS settings. If you want to map an existing domain name to your Amazon EC2 instance running Windows Server, use a DNS service provider such as Amazon Route 53. When using your own domain name, we recommend that you map to the instance's external DNS name using a CNAME, not by using an A record that points to the instance's IP address.

To map your own domain name to your CloudFront distribution, see Using CNAMEs (p. 49).

# How can I enable access to the Windows server?

**To enable access to port 3389 on your Windows server via selected IP addresses**

By default, the Amazon EC2 security group for your Windows server instance does not have port 3389 enabled; this is the port you use to administer the Windows server. If you want to log on to your Windows server instance, perform the following procedure to enable access via port 3389.

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Navigation** pane, in the **Region** list, click the Amazon EC2 region in which you used AWS CloudFormation to create your Amazon EC2 instance.
3. In the **Navigation** pane, click **Security Groups**.
4. In the **Security Groups** pane, click the row for which the value of the **Name** column begins with the name of the AWS CloudFormation stack that you created in Creating an AWS CloudFormation Stack for Live Smooth Streaming (p. 163).
5. In the bottom pane, click the **Inbound** tab.
6. To enable access to your Windows server and specify the client IP addresses that can access the server:

   a. In the **Create a new rule** list, do not change the default value, **Custom TCP rule**.
   b. In the **Port range** field, enter **3389**.
   c. In the **Source** field, enter an IP address or range, or enter the name of another security group. For more information, click **Help**.
   d. Click **Add Rule**.
   e. To create additional rules, repeat Steps a through d.
   f. Click **Apply Rule Changes**.

# How can I securely connect to my Amazon EC2 instance running Windows IIS Media Services?

To connect to your Windows server instance, you must retrieve the initial password for the Administrator account and then use it with Windows Remote Desktop. You'll also need the contents of the private key file that you created, for example, *<keypairname.pem>*.pem. For more information, go to Getting Started Guide AWS Computing Basics for Windows.

# How can I restrict access to my Live Smooth Streaming content from another domain?

Microsoft Silverlight includes support for cross-domain connectivity, which allows the Silverlight player to access content from locations other than the domain where the Smooth Streaming content originates. The security policy system in Silverlight requires that a Silverlight policy file named `ClientAccessPolicy.xml` be downloaded from a target domain before a network connection is allowed to access a network resource under that target domain. A default policy file is already included at the root of the default website on your Windows server running on Amazon EC2. To restrict cross-domain access, log on to your Windows server and update the `ClientAccessPolicy.xml` file.

# Additional Documentation

## Microsoft Documentation

- IIS Smooth Streaming Deployment Guide
- IIS Media Services 4.1 Readme
- IIS Smooth Streaming Management REST Services
- Configuring Authentication in IIS 7
- Microsoft Expression Encoder blog
- Managing live publishing points from Microsoft Expression Encoder 4 Pro SP2
- Live IIS Smooth Streaming in Expression Encoder 4 Pro
- Apple HTTP Live Streaming with IIS Media Services

## Amazon Web Services Documentation

- Amazon EC2 Running Microsoft Windows Server & SQL Server
- Amazon Elastic Compute Cloud Microsoft Windows Guide
- Amazon CloudFront
- AWS CloudFormation

# Restricting Access to Files in a CloudFront Distribution Based on Geographic Location (Geoblocking)

**Topics**

## Overview of Restricting Access to Files in a CloudFront Distribution Based on Geographic Location

Amazon CloudFront improves the performance, reliability, and availability of your websites and applications by distributing your web content, such as images, video, and audio to a worldwide network of edge locations. When an end user requests your content, CloudFront serves your content to the user from the edge location that has the lowest latency for that user at that moment. If you have geographic restrictions on where your content can be distributed, you can use CloudFront with a third-party geolocation service to control distribution of your content according to the location of a request. This is known as geoblocking or geotargeting. For example, if a request comes from a country where, for copyright reasons, you are not authorized to distribute your content, you can block the request and direct the requester to a message that explains the situation.

Here's how it works:

1. An end user who is viewing your website requests a web page or a file that is georestricted.

2. Your web application gets the end user's IP address from the request and sends the IP address to a geolocation service. You will need an account with one of these services.

3. The geolocation service determines the geographic location of the end user's IP address and returns the result to your web application.

4. Your web application compares the end user's location with a list of locations where the file can (or can't) be distributed:

   - If the end user is allowed to access the web page or file, your application creates a CloudFront signed URL and returns it to the end user.

   - If the end user is not allowed to access the web page or file, your web application returns the URL of a "you are not authorized" message to the end user.

5. If the end user is allowed to access the web page or file, the end user's browser automatically uses the signed URL to request the file from CloudFront.

Amazon CloudFront
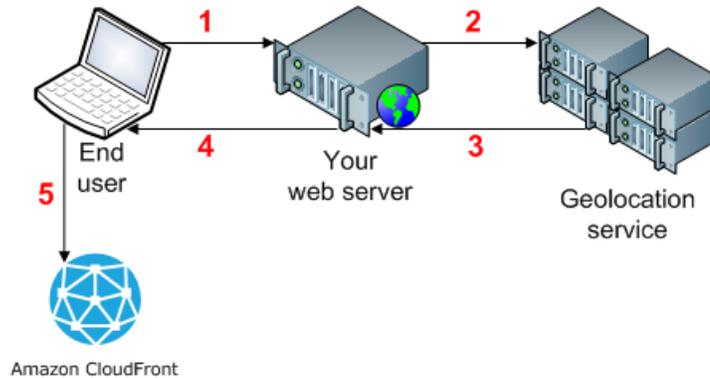
Using CloudFront and a third-party geolocation service to restrict access to your content from your application layer gives you full control over your end user's experience. For end users whose access is blocked, your application can display a meaningful message instead of returning an error code. You can also customize the error message you display for your end users according to their location.

The following task list guides you through the process of implementing geoblocking functionality in your applications to restrict access to the content in your CloudFront distribution according to the end user's location.

**Task list for restricting access to files in a CloudFront distribution based on geographic location**

1.  Get an account with a geolocation service.

    This section provides sample code for Digital Element and for MaxMind, but any geolocation service is supported.

2.  If you don't already have an AWS account, create one. For more information, see Creating an Amazon Web Services Account (p. 180)

3.  Upload your content to an Amazon Simple Storage Service (S3) bucket. For more information, see the Amazon S3 documentation.

4.  Configure Amazon CloudFront and Amazon S3 to serve private content. For more information, see Private Content Process Overview.

5.  Write your web application to do the following:

    a.  Send the IP address for each end-user request to the geolocation service.

    b.  Evaluate the return value from the geolocation service (commonly a country code) to determine whether the end user is in a location to which you want CloudFront to distribute your content.

    c.  Either generate a signed URL for your CloudFront content, or block access to the content.


    Java, .NET, and PHP sample code is provided below for Digital Element and for MaxMind. See the applicable topic:

    *   Sample Code for Digital Element (p. 180)
    *   Sample Code for MaxMind (p. 190)


    If you're using another geolocation service, refer to their documentation.


Amazon Web Services provides SDKs for Java, .NET, and PHP. For more information, see the applicable page on the Amazon Web Services website:

- Java Developer Center

- Windows & .NET Developer Center

- PHP Developer Center

# Creating an Amazon Web Services Account

> **Note**
>
> When you create an account, AWS automatically signs up the account for all services. You are charged only for the services you use.

**To create an AWS account**

1. Go to http://aws.amazon.com, and click **Create an AWS Account**.
2. Follow the on-screen instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.

# Sample Code for Digital Element

The samples in this section show how to get a location from Digital Element from an end user's IP address. The samples also show how to create a signed URL for a requested object if you are allowed to distribute the content to the end user's location.

All sample code was tested before the document was published, but subsequent changes to the Digital Element API could affect whether the samples are still accurate. For the latest information, go to the Digital Element documentation.

See the applicable sample code:

- Java Sample Code for Digital Element (p. 180)
- .NET Sample Code for Digital Element (p. 185)
- PHP Sample Code for Digital Element (p. 188)

> **Note**
>
> In the code examples, red italicized text is a placeholder. Replace this text with whatever values are appropriate for your situation.

# Java Sample Code for Digital Element

The code example provided here obtains the country code that is associated with an end user's IP address and allows the user to access CloudFront content if the user is in a location where distribution is allowed. For the purposes of the example, the program is authorized to distribute the requested content to any country except Australia (country code AU).

## GetCountryCodeServlet.java

GetCountryCodeServlet.java calls GetDigitalElementCountryCode.java, which is shown later in this article, to ask Digital Element for the country code that is associated with an end user's IP address. If the country code is not AU (Australia), GetCountryCodeServlet.java calls SignedUrl.java to create a signed URL that the end user can use to access a file in the CloudFront distribution.

```
/*
 * Copyright 2011 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 *   http://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 */

// Signed URLs for a private distribution
// Note that Java supports SSL certificates only in DER format,
// so you will need to convert your PEM-formatted file to DER format.
// To do this, you can use openssl:
// openssl pkcs8 -topk8 -nocrypt -in origin.pem -inform PEM -out new.der -outform
 DER
// For the encoder to work correctly, you should also add the
// bouncy castle jar to your project and then add the provider.ds.

import java.io.IOException;
import java.io.PrintWriter;
import java.util.StringTokenizer;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class GetCountryCodeServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    final String GEOAPIURL = "Digital Element URL";
    final String GEOAPITOKEN = "Digital Element user token";
    final String PATHTODER = "path to .der file";
    final String KEYPAIRID ="CloudFront key pair ID";
    final String HTTPORHTTPS = "https";
    final String CFDISTRIBUTION = "dxxxx.cloudfront.net";
    final String CFPATH = "CloudFront URL for file";
    // date and time that CloudFront's signed URL expires,
    // in Coordinated Universal Time
    final String EXPIRETS = "2012-11-14T22:20:00.000Z";
    final String BLOCKEDCOUNTRY="AU";

  protected void doGet(HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException {
```

```java
        String ip = null;
        StringTokenizer st = null;
        PrintWriter out = response.getWriter();

        String headers = request.getHeader("X-FORWARDED-FOR");

        if (headers!= null){
            st = new StringTokenizer(headers,",");

            while (st.hasMoreTokens()) {
                ip = st.nextToken();
            }
        }

        //Get the client's IP addr in case X-Forwarded-IP header doesn't exist

        if (ip == null) ip = request.getRemoteAddr();

        try {
          GetDigitalElementCountryCode country = new GetDigitalElementCountryCode(
 GEOAPIURL,GEOAPITOKEN );

            if ( !country.getCountry(ip).equalsIgnoreCase(BLOCKEDCOUNTRY)){

                SignedUrl myApp = new SignedUrl(KEYPAIRID,PATHTODER);
              out.println(myApp.getSignedHash(HTTPORHTTPS,CFDISTRIBUTION,CFPATH,EX
PIRETS));

            }else {
                out.println("You cannot access this link.");
            }
        } catch (Exception e1) {
          e1.printStackTrace();
        }
    }
}
```

## GetDigitalElementCountryCode.java

GetDigitalElementCountryCode.java sends Digital Element a request that includes an end user's IP address. The return value is a country code.

```java
/*
 * Copyright 2011 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 *  http://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 */
```

```
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;

public class GetDigitalElementCountryCode {

    private static String geoApiEndPoint;
    private static String apiToken;

    GetDigitalElementCountryCode(String mygeoApiEndPoint, String myapiToken){
        geoApiEndPoint = mygeoApiEndPoint;
        apiToken = myapiToken;
    }

     public String getCountry(String enduserIP) throws Exception {

        String geoApiURL = "http://"+geoApiEndPoint+"?u="+apiToken+"&ip="+end
userIP;

        DocumentBuilderFactory docBuilderFactory = DocumentBuilderFactory.newIn
stance();
        DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder();
        Document doc = docBuilder.parse(geoApiURL);
        // normalize text representation
        doc.getDocumentElement ().normalize ();

        NodeList listOfPersons = doc.getElementsByTagName("response");
        Element el = (Element)listOfPersons.item(0);
        String country = el.getAttribute("edge-two-letter-country");

        return country;
    }
}
```

## SignedUrl.java

SignedUrl.java creates a signed URL that the end user can use to access a file in the CloudFront
distribution.

```
/*
 * Copyright 2011 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 *  http://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 */
```

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.security.Security;
import java.text.ParseException;

import org.jets3t.service.CloudFrontService;
import org.jets3t.service.CloudFrontServiceException;
import org.jets3t.service.utils.ServiceUtils;

public class SignedUrl {
    // Signed URLs for a private distribution
    // Note that Java supports SSL certificates only in DER format,
    // so you need to convert your PEM-formatted file to DER format.
    // To do this, you can use openssl:
    // openssl pkcs8 -topk8 -nocrypt -in origin.pem -inform PEM -out new.der -
outform DER
    // For the encoder to work correctly, you should also add the
    // bouncy castle jar to your project and then add the provider.ds.

    private static String keyPairId;
    private static String privateKeyFilePath;

    SignedUrl(String mykeyPairId, String myprivateKeyFilePath){
        keyPairId = mykeyPairId;
        privateKeyFilePath = myprivateKeyFilePath;
    }

    public String getSignedHash(String protocol, String cfDistribution, String
objectUri, String expTime) throws FileNotFoundException, IOException,
    CloudFrontServiceException, ParseException{

    Security.addProvider(new org.bouncycastle.jce.provider.BouncyCastlePro
vider());

    // Convert your DER file into a byte array.

        byte[] derPrivateKey = ServiceUtils.readInputStreamToBytes(new FileInput
Stream(privateKeyFilePath));

    // Generate a "canned" signed URL to allow access to a
    // specific distribution and object

    String signedUrlCanned = CloudFrontService.signUrlCanned(
        protocol+ "://" + cfDistribution + "/" + objectUri, // Resource URL or
Path
        keyPairId,     // Certificate identifier,
                       // an active trusted signer for the distribution
        derPrivateKey, // DER Private key data
        ServiceUtils.parseIso8601Date(expTime) // DateLessThan
        );

    return signedUrlCanned;
    }
}
```

# .NET Sample Code for Digital Element

The following sample application gets the IP address of the end user and sends the IP address to Digital Element. Digital Element returns the country code (in XML format) that corresponds with the end user's IP address. The application parses the XML and evaluates whether the value returned by Digital Element matches the blocked country code. If the end user's country is blocked, the application displays a message to that effect. If the end user's country is not blocked, the application creates a signed URL that expires in one minute, performs the substitutions necessary to ensure that the URL doesn't include any invalid characters, and redirects the user's browser to the signed URL.

```
<%@ Page Language="C#" AutoEventWireup="true"  %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "ht
tp://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
    <%=GetContent()%>
    </div>
    </form>
</body>
</html>

<%@ Import Namespace="System.Linq" %>
<%@ Import Namespace="System.Xml.Linq" %>
<%@ Import Namespace="System.Security.Cryptography" %>
<%@ Import Namespace="System.Net" %>
<%@ Import Namespace="System.IO" %>

<script runat="server">

    // Key pair ID for the CloudFront key pair
    private const string KEYPAIR_ID = "CloudFront key pair ID";

    // Private key for the CloudFront key pair.
    // The value is derived from opensslkey.
    private const string PRIVATE_KEY = "private key";

    // JSON policy statement used in the expiring URL
   private const string POLICY = "{{\"Statement\":[{{\"Resource\":\"{0}\",\"Con
dition\":{{\"DateLessThan\":{{\"AWS:EpochTime\":{1}}}}}}}}]}}";

    // Digital Element user token to be passed to geolocation service call

    private const string USERTOKEN = "Digital Element user token";
    private const string GEOAPIURL = "Digital Element URL";

    // GEO IP service URL with parameters:
    // {0} = User Token and {1} = IP Address
    private const string SERVICEURL = GEOAPIURL + "?u={0}&ip={1}";

    // Array of countries to block
```

```csharp
    private static readonly string[] COUNTRIES_TO_BLOCK = new String[] {"US"};

    private const string BLOCKED_MSG = "Your access to this content is blocked
because you're visiting from '{0}'.";

    /// <summary>
    /// Returns the IP address coming from the request object.
    /// </summary>
    /// <returns>The IP address for the request.</returns>
    private string GetOriginIpAddress()
    {
        // .NET provides Request.UserHostAddress to get the
        // remote IP address, but this could be the IP address of the
        // last proxy in a chain, for example, an Elastic Load Balancer.
        // Instead, use the HTTP_X_FORWARDED_FOR header if one exists.
        string forwardedIpAddresses = this.Request.ServerVariables["HTTP_X_FORWAR
DED_FOR"];

        if (string.IsNullOrEmpty(forwardedIpAddresses))
        {
            // Simply return the UserHostAddress.
            return Request.UserHostAddress;
        }
        else
        {
            // Get the last item in the list.
            return forwardedIpAddresses.Split(',').Last().Trim();
        }
    }

    /// <summary>
    /// This function returns the country code
    /// associated with the IP address in the request object.
    /// </summary>
    /// <returns>The country code for the request.</returns>
    private string GetCountryCodeFromIP()
    {
        var ipAddress = GetOriginIpAddress();
        var serviceURL = String.Format(SERVICEURL, Server.UrlEncode(USERTOKEN),
Server.UrlEncode(ipAddress));

        try
        {
            var xDoc = XDocument.Load(serviceURL);
            var res = (from w in xDoc.Descendants("response") select w).First();

            return res.Attribute("edge-two-letter-country").Value.ToUpper();
        }
        catch(Exception ex)
        {
            // There was an error in making the web request.
            this.Response.Write(serviceURL +  "<br><br>");
            this.Response.Write(ex.Message);
            this.Response.End();
        }
        return null;
    }
```

```
    /// <summary>
    /// This function returns a signed URL that will expire in 1 minute.
    /// For more information, see "Create a URL Signature Using C# and the
    /// .NET Framework" in the Amazon CloudFront Developer Guide:
    /// http://docs.amazonwebservices.com/AmazonCloudFront/latest/Developer
Guide/CreateSignatureInCSharp.html?r=4472
    /// </summary>
    /// <param name="resourceUrl"></param>
    /// <returns></returns>
    private string GetSignedURL(string resourceUrl)
    {
        // Compute expiration date.
        var endTimeSpanFromNow = new TimeSpan(0, 1, 0);
        var intervalEnd = (DateTime.UtcNow.Add(endTimeSpanFromNow)) - new Date
Time(1970, 1, 1, 0, 0, 0, DateTimeKind.Utc);
        var endTimestamp = (int)intervalEnd.TotalSeconds; // Timestamp must be a
 whole number
        var expires = endTimestamp.ToString();
        var strPolicy = string.Format(POLICY, resourceUrl, expires);

        // Encrypt the policy.
        var bufferPolicy = Encoding.ASCII.GetBytes(strPolicy);
        var cryptoSHA1 = new SHA1CryptoServiceProvider();
        bufferPolicy = cryptoSHA1.ComputeHash(bufferPolicy);
        var providerRSA = new RSACryptoServiceProvider();
        providerRSA.FromXmlString(PRIVATE_KEY);
        var rsaFormatter = new RSAPKCS1SignatureFormatter(providerRSA);
        rsaFormatter.SetHashAlgorithm("SHA1");
        var signedPolicyHash = rsaFormatter.CreateSignature(bufferPolicy);
        var strSignedPolicy = System.Convert.ToBase64String(signedPolicyHash);

        // Build the query string with the expiration, policy signature,
        // and CloudFront key pair ID.
        var queryString = "Expires={0}&Signature={1}&Key-Pair-Id={2}";
        queryString = String.Format(queryString, Server.UrlEncode(expires),
Server.UrlEncode(strSignedPolicy), Server.UrlEncode(KEYPAIR_ID));
        var urlString = resourceUrl + "?" + queryString;
        return urlString;
    }

    /// <summary>
    /// Return a message saying this is blocked because of your country, or
    /// return an image tag.
    /// </summary>
    /// <returns></returns>
    public string GetContent()
    {
        var country = GetCountryCodeFromIP();
        if (COUNTRIES_TO_BLOCK.Contains(country))
        {
            // The country returned from the call to the geolocation service
            // is listed in the array of blocked countries.
            return string.Format(BLOCKED_MSG, country);
        }
        else
        {
            // The country returned from the call to the geolocation service
            // is NOT listed in the array of blocked countries
```

```
            // Get a CloudFront signed URL for the content and display it.
            var url = GetSignedURL("CloudFront URL");
            var img = "<img src='{0}' />";
            return String.Format(img, url);
        }
    }
</script>
```

# PHP Sample Code for Digital Element

The following sample application gets the IP address of the end user and sends the IP address to Digital Element. Digital Element returns the country code (in XML format) that corresponds to the end user's IP address. The application then parses the XML, displays the country code that is blocked, and evaluates whether the value returned by Digital Element matches the blocked country code. If the end user's country is not blocked, the application displays a "You are not blocked" message, uses a canned policy to create a signed URL that expires in five minutes, performs the substitutions necessary to ensure that the URL doesn't include any invalid characters, and redirects the user's browser to the signed URL. If the end user's country is blocked, the application displays a "You are blocked" message and a graphic.

```
<!DOCTYPE html>
<html>
<head>
    <title>Geoblocking Test</title>
</head>
<body>
    <h1>Geoblocking Test</h1>

<?php
// Configure the private key (make sure this information is secure).
$private_key_filename = 'path to private key';
$key_pair_id          = 'CloudFront key pair ID';

/*
 * Configure the geoblocking parameters. The following variables
 * describe the two-letter country to be blocked, the
 * CloudFront URL for the file that you want to secure,
 * and the expiry time of the URL. Change these values as needed.
 */
$blocked_geo = 'uk';
$asset_path  = 'CloudFront URL for the object';
$expires     = time() + 300; // (5 minutes from now)

// Configure the URL to the geoblocking service.
$token       = 'Digital Element user token';
$address     = 'Digital Element URL';
$remote_ip   = get_remote_ip_address();
$service_url = $address . '?u=' . $token . '&ip=' . $remote_ip;

// Call the web service using the configured URL.
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $service_url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
$ws_response = curl_exec($ch);

// Parse the response with SimpleXML and get the geoblocking value.
$xml        = new SimpleXMLElement($ws_response);
```

```php
$edge_geo = $xml->response->attributes()->{'edge-two-letter-country'};

echo '<p>The country being blocked is: ' . strtoupper($blocked_geo) . '</p>';

if ($edge_geo != $blocked_geo)
{
   echo '<p>Your country is: ' . strtoupper($edge_geo) . '</p>';
   echo '<p>You are not blocked.</p>';
   $signed_url = create_signed_url($asset_path, $private_key_filename,
$key_pair_id, $expires);
   echo '<img src="' . $signed_url . '" width="600" height="401" ' ;
}
else
{
   echo '<p>Your country is: ' . strtoupper($edge_geo) . '</p>';
   echo '<p>You are blocked.</p>';
  $blocked_url = 'http://s3.amazonaws.com/<Amazon S3 bucket>/blocked-image.jpg';

   echo '<img src="' . $blocked_url . '" alt="Access blocked" width="600"
height="401" ';
}

// Function definitions


function get_remote_ip_address()
{
   // Check to see if an HTTP_X_FORWARDED_FOR header is present.

   if($_SERVER["HTTP_X_FORWARDED_FOR"])

   {

      // If the header is present, use the last IP address.
      $temp_array     = explode(',', $_SERVER['HTTP_X_FORWARDED_FOR']);
      $temp_ip_address = $temp_array[count($temp_array) - 1];
   }
   else
   {
      // If the header is not present, use the
      // default server variable for remote address.
      $temp_ip_address = $_SERVER['REMOTE_ADDR'];
   }

   return $temp_ip_address;
}

function create_signed_url($asset_path, $private_key_filename, $key_pair_id,
$expires)
{
   // Build the policy.
   $canned_policy = '{"Statement":[{"Resource":"' . $asset_path
      . '","Condition":{"DateLessThan":{"AWS:EpochTime":'. $expires . '}}}]}';


   // Sign the policy.
   $signature = rsa_sha1_sign($canned_policy, $private_key_filename);
```

```php
    // Make the signature is safe to be included in a URL.
    $encoded_signature = url_safe_base64_encode($signature);

    // Combine the above into a properly formed URL name.
    $temp_signed_url = $asset_path . '?Expires=' . $expires . '&Signature='
        . $encoded_signature . '&Key-Pair-Id=' . $key_pair_id;

    return $temp_signed_url;
}

function rsa_sha1_sign($policy, $private_key_filename)
{
    $signature = '';

    // Load the private key.
    $fp = fopen($private_key_filename, 'r');
    $private_key = fread($fp, 8192);
    fclose($fp);

    $private_key_id = openssl_get_privatekey($private_key);

    // Compute the signature.
    openssl_sign($policy, $signature, $private_key_id);

    // Free the key from memory.
    openssl_free_key($private_key_id);

    return $signature;
}

function url_safe_base64_encode($value)
{
    $encoded = base64_encode($value);

    // Replace the characters that cannot be used in a URL.
    return str_replace(array('+', '=', '/'), array('-', '_', '~'), $encoded);
}
?>

</body>
</html>
```

# Sample Code for MaxMind

The samples in this section show how to get a location from MaxMind from an end user's IP address and, if you are authorized to distribute the requested object to the user's location, how to create a signed URL for the object.

All sample code was tested before the document was published, but subsequent changes to the MaxMind API could affect whether the samples are still accurate. For the latest information, go to the MaxMind documentation.

See the applicable sample code:

# Java Sample Code for MaxMind

## GetCountryCodeServlet.java

GetCountryCodeServlet.java calls GetMaxMindCountryCode.java, which is shown later in this article, to ask MaxMind for the country code that is associated with an end user's IP address. If the country code is not AU (Australia), GetCountryCodeServlet.java calls SignedUrl.java to create a signed URL that the end user can use to access a file in the CloudFront distribution.

```java
/*
 * Copyright 2011 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 *   http://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 */

// Signed URLs for a private distribution
// Note that Java supports SSL certificates only in DER format,
// so you will need to convert your PEM-formatted file to DER format.
// To do this, you can use openssl:
// openssl pkcs8 -topk8 -nocrypt -in origin.pem -inform PEM -out new.der -outform
 DER
// For the encoder to work correctly, you should also add the
// bouncy castle jar to your project and then add the provider.ds.

import java.io.IOException;
import java.io.PrintWriter;
import java.util.StringTokenizer;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class GetCountryCodeServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    final String GEOAPIURL = "MaxMind URL";
    final String GEOAPITOKEN = "MaxMind user token";
    final String PATHTODER = "path to .der file";
    final String KEYPAIRID ="CloudFront key pair ID";
    final String HTTPORHTTPS = "https";
    final String CFDISTRIBUTION = "dxxxx.cloudfront.net";
    final String CFPATH = "CloudFront URL for file";
    // date and time that CloudFront's signed URL expires,
    // in Coordinated Universal Time
```

```
   final String EXPIRETS = "2012-11-14T22:20:00.000Z";
   final String BLOCKEDCOUNTRY="AU";

  protected void doGet(HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException {

      String ip = null;
      StringTokenizer st = null;
      PrintWriter out = response.getWriter();

      String headers = request.getHeader("X-FORWARDED-FOR");

      if (headers!= null){
         st = new StringTokenizer(headers,",");

         while (st.hasMoreTokens()) {
            ip = st.nextToken();
         }
      }

      //Get the client's IP addr in case X-Forwarded-IP header doesn't exist.

      if (ip == null) ip = request.getRemoteAddr();

      try {

         GetMaxMindCountryCode country  = new GetMaxMindCountryCode("GEOAPI
URL","GEOAPITOKEN");

         if ( !country.getCountry(ip).equals(BLOCKEDCOUNTRY)){

            SignedUrl myApp = new SignedUrl(KEYPAIRID,PATHTODER);
          out.println(myApp.getSignedHash(HTTPORHTTPS,CFDISTRIBUTION,CFPATH,EX
PIRETS));

         }else {
            out.println("You cannot access this link.");
         }
      } catch (Exception e1) {
         e1.printStackTrace();
      }
   }
}
```

## GetMaxMindCountryCode.java

GetMaxMindCountryCode.java sends MaxMind a request that includes an end user's IP address. The
return value is a country code.

```
/*
 * Copyright 2011 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 *  http://aws.amazon.com/apache2.0
```

```
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 */

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URL;
import java.net.URLConnection;

public class GetMaxMindCountryCode {

 private static String geoApiEndPoint;
 private static String apiToken;

 GetMaxMindCountryCode(String mygeoApiEndPoint, String myapiToken){
  geoApiEndPoint = mygeoApiEndPoint;
  apiToken = myapiToken;
 }

 public String getCountry(String enduserIP) throws Exception {
  String geoApiURL = "http://"+geoApiEndPoint+"?l="+apiToken+"&i="+enduserIP;

  // Call to MaxMind API.
  URL url = new URL(geoApiURL);
  URLConnection urlConn = url.openConnection();

  urlConn.setUseCaches(false);

  InputStreamReader in = new InputStreamReader((InputStream) urlConn.getCon
tent());
  BufferedReader buff = new BufferedReader(in);

  return buff.readLine();
 }
}
```

## SignedUrl.java

SignedUrl.java creates a signed URL that the end user can use to access a file in the CloudFront
distribution.

```
/*
 * Copyright 2011 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 *  http://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
```

```
 * permissions and limitations under the License.
 */

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.security.Security;
import java.text.ParseException;

import org.jets3t.service.CloudFrontService;
import org.jets3t.service.CloudFrontServiceException;
import org.jets3t.service.utils.ServiceUtils;

public class SignedUrl {
    // Signed URLs for a private distribution
    // Note that Java supports SSL certificates only in DER format,
    // so you need to convert your PEM-formatted file to DER format.
    // To do this, you can use openssl:
    // openssl pkcs8 -topk8 -nocrypt -in origin.pem -inform PEM -out new.der -
outform DER
    // For the encoder to work correctly, you should also add the
    // bouncy castle jar to your project and then add the provider.ds.

    private static String keyPairId;
    private static String privateKeyFilePath;

    SignedUrl(String mykeyPairId, String myprivateKeyFilePath){
        keyPairId = mykeyPairId;
        privateKeyFilePath = myprivateKeyFilePath;
    }

    public String getSignedHash(String protocol, String cfDistribution, String
objectUri, String expTime) throws FileNotFoundException, IOException,
    CloudFrontServiceException, ParseException{

    Security.addProvider(new org.bouncycastle.jce.provider.BouncyCastlePro
vider());

    // Convert your DER file into a byte array.

        byte[] derPrivateKey = ServiceUtils.readInputStreamToBytes(new FileInput
Stream(privateKeyFilePath));

    // Generate a "canned" signed URL to allow access to a
    // specific distribution and object.

    String signedUrlCanned = CloudFrontService.signUrlCanned(
        protocol+ "://" + cfDistribution + "/" + objectUri, // resource URL or
path
        keyPairId,     // Certificate identifier,
                        // an active trusted signer for the distribution
        derPrivateKey, // DER private key data
        ServiceUtils.parseIso8601Date(expTime) // DateLessThan
        );

    return signedUrlCanned;
    }
}
```

# PHP Sample Code for MaxMind

The following sample application gets the IP address of the end user and sends the IP address to MaxMind. MaxMind returns the country code that corresponds to the end user's IP address. The application then displays the country code that is blocked and evaluates whether the value returned by MaxMind matches the blocked country code. If the end user's country is not blocked, the application displays a "You are not blocked" message, uses a canned policy to create a signed URL that expires in five minutes, performs the substitutions necessary to ensure that the URL doesn't include any invalid characters, and redirects the user's browser to the signed URL. If the end user's country is blocked, the application displays a "You are blocked" message and a graphic.

```php
<!DOCTYPE html>
<html>
<head>
    <title>Geoblocking Test</title>
</head>
<body>
    <h1>Geoblocking Test</h1>

<?php
// Configure the private key (make sure this information is secure).
$private_key_filename = 'path to private key';
$key_pair_id          = 'CloudFront key pair ID';

/*
 * Configure the geoblocking parameters. The following variables
 * describe the two-letter country to be blocked, the
 * CloudFront URL for the file that you want to secure,
 * and the expiry time of the URL. Change these values as needed.
 */
$blocked_geo = 'gb';
$asset_path  = 'CloudFront URL for the object';
$expires     = time() + 300; // (5 minutes from now)

// Configure the URL to the geolocation service.
$token       = 'MaxMind user token';
$address     = 'MaxMind URL';
$remote_ip   = get_remote_ip_address();
$service_url = $address . '?l=' . $token . '&i=' . $remote_ip;

// Call the web service using the configured URL.
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $service_url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
$ws_response = curl_exec($ch);

$edge_geo = $ws_response;

echo '<p>The country being blocked is: ' . strtoupper($blocked_geo) . '</p>';

if ($edge_geo != strtoupper($blocked_geo))
{
    echo '<p>Your country is: ' . strtoupper($edge_geo) . '</p>';
    echo '<p>You are not blocked.</p>';
    $signed_url = create_signed_url($asset_path, $private_key_filename,
$key_pair_id, $expires);
    echo '<img src="' . $signed_url . '" width="600" height="401" />' ;
```

```
}
else
{
   echo '<p>Your country is: ' . strtoupper($edge_geo) . '</p>';
   echo '<p>You are blocked.</p>';
   $blocked_url = 'http://s3.amazonaws.com/<Amazon S3 bucket>/blocked-image.jpg';

   echo '<img src="' . $blocked_url . '" alt="Access blocked" width="600"
height="401" />';
}

// Function definitions

function get_remote_ip_address()
{
   // Check to see if an HTTP_X_FORWARDED_FOR header is present.

   if($_SERVER['HTTP_X_FORWARDED_FOR'])

   {

      // If the header is present, use the last IP address.
      $temp_array     = explode(',', $_SERVER['HTTP_X_FORWARDED_FOR']);
      $temp_ip_address = $temp_array[count($temp_array) - 1];
   }
   else
   {
      // If the header is not present, use the
      // default server variable for remote address.
      $temp_ip_address = $_SERVER['REMOTE_ADDR'];
   }

   return $temp_ip_address;
}


function create_signed_url($asset_path, $private_key_filename, $key_pair_id,
$expires)
{
   // Build the policy.
   $canned_policy = '{"Statement":[{"Resource":"' . $asset_path
      . '","Condition":{"DateLessThan":{"AWS:EpochTime":'. $expires . '}}}]}';


   // Sign the policy.
   $signature = rsa_sha1_sign($canned_policy, $private_key_filename);

   // Make the signature contains only characters that
   // can be included in a URL.
   $encoded_signature = url_safe_base64_encode($signature);

   // Combine the above into a properly formed URL name
   $temp_signed_url = $asset_path . '?Expires=' . $expires . '&Signature='
      . $encoded_signature . '&Key-Pair-Id=' . $key_pair_id;

   return $temp_signed_url;
}
```

```
function rsa_sha1_sign($policy, $private_key_filename)
{
    $signature = '';

    // Load the private key.
    $fp = fopen($private_key_filename, 'r');
    $private_key = fread($fp, 8192);
    fclose($fp);

    $private_key_id = openssl_get_privatekey($private_key);

    // Compute the signature.
    openssl_sign($policy, $signature, $private_key_id);

    // Free the key from memory.
    openssl_free_key($private_key_id);

    return $signature;
}

function url_safe_base64_encode($value)
{
    $encoded = base64_encode($value);

    // Replace characters that cannot be included in a URL.
    return str_replace(array('+', '=', '/'), array('-', '_', '~'), $encoded);
}
?>

</body>
</html>
```

# .NET Sample Code for MaxMind

The following sample application gets the IP address of the end user and sends the IP address to MaxMind. MaxMind returns the country code that corresponds with the end user's IP address. The application then evaluates whether the value returned by Digital Element matches the blocked country code. If the end user's country is blocked, the application displays a message to that effect. If the end user's country is not blocked, the application creates a signed URL that expires in one minute, performs the substitutions necessary to ensure that the URL doesn't include any invalid characters, and redirects the user's browser to the signed URL.

```
<%@ Page Language="C#" AutoEventWireup="true"  %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "ht
tp://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
    <%=GetContent()%>
    </div>
```

```
      </form>
</body>
</html>

<%@ Import Namespace="System.Linq" %>
<%@ Import Namespace="System.Xml.Linq" %>
<%@ Import Namespace="System.Security.Cryptography" %>
<%@ Import Namespace="System.Net" %>
<%@ Import Namespace="System.IO" %>

<script runat="server">

   // Key pair ID for the CloudFront key pair
   private const string KEYPAIR_ID = "CloudFront key pair ID";

   // Private key for the CloudFront key pair.
   // The value is derived from opensslkey.
   private const string PRIVATE_KEY = "private key";

   // JSON policy statement used in the expiring URL
   private const string POLICY = "{{\"Statement\":[{{\"Resource\":\"{0}\",\"Con
dition\":{{\"DateLessThan\":{{\"AWS:EpochTime\":{1}}}}}}}}]}}";

   // User token to be passed in to GEO IP service call
   private const string USERTOKEN = "user token";

   // Geolocation service URL with parameters:
   // {0} = User Token and {1} = IP address
   private const string SERVICEURL = "http://geoip3.maxmind.com/a?l={0}&i={1}";


   // Array of countries to block
   private static readonly string[] COUNTRIES_TO_BLOCK = new String[] {"US"};

   private const string BLOCKED_MSG = "Your access to this content is blocked
because you're visiting from '{0}'.";

   /// <summary>
   /// Returns the IP address coming from the request object.
   /// </summary>
   /// <returns>The IP address for the request.</returns>
   private string GetOriginIpAddress()
   {
      // .NET provides Request.UserHostAddress to get the
      // remote IP address, but this could be the IP address of the
      // last proxy in a chain, for example, an Elastic Load Balancer.
      // Instead use the HTTP_X_FORWARDED_FOR header if one exists.
      string forwardedIpAddresses = this.Request.ServerVariables["HTTP_X_FORWAR
DED_FOR"];

      if (string.IsNullOrEmpty(forwardedIpAddresses))
      {
         // Return the UserHostAddress.
         return Request.UserHostAddress;
      }
      else
      {
         // Get the last item in the list.
```

```csharp
            return forwardedIpAddresses.Split(',').Last().Trim();
        }
    }

    /// <summary>
    /// This function returns the country code
    /// associated with the IP address in the request object.
    /// </summary>
    /// <returns>The country code for the request.</returns>
    private string GetCountryCodeFromIP()
    {
        var ipAddress = GetOriginIpAddress();
        var serviceURL = String.Format(SERVICEURL, Server.UrlEncode(USERTOKEN),
Server.UrlEncode(ipAddress));

        try
        {
            var webReq = HttpWebRequest.Create(serviceURL);
            var webRes = webReq.GetResponse().GetResponseStream();
            var sr = new StreamReader(webRes);
            var strRes = sr.ReadToEnd();
            sr.Close();
            return strRes.Trim().ToUpper();
        }
        catch(Exception ex)
        {
            // There was an error in making the web request.
            this.Response.Write(serviceURL +  "<br><br>");
            this.Response.Write(ex.Message);
            this.Response.End();
        }
        return null;
    }

    /// <summary>
    /// This function returns a signed URL that will expire
    /// in 1 minute. For more information, see "Create a URL Signature
    /// Using C# and the .NET Framework" in the Amazon CloudFront Developer
Guide:
    /// http://docs.amazonwebservices.com/AmazonCloudFront/latest/Developer
Guide/CreateSignatureInCSharp.html?r=4472
    /// </summary>
    /// <param name="resourceUrl"></param>
    /// <returns></returns>
    private string GetSignedURL(string resourceUrl)
    {
        // Compute expiration date.
        var endTimeSpanFromNow = new TimeSpan(0, 1, 0);
        var intervalEnd = (DateTime.UtcNow.Add(endTimeSpanFromNow)) - new Date
Time(1970, 1, 1, 0, 0, 0, DateTimeKind.Utc);
        var endTimestamp = (int)intervalEnd.TotalSeconds; // Timestamp must be a
 whole number
        var expires = endTimestamp.ToString();
        var strPolicy = string.Format(POLICY, resourceUrl, expires);

        // Encrypt the policy.
        var bufferPolicy = Encoding.ASCII.GetBytes(strPolicy);
        var cryptoSHA1 = new SHA1CryptoServiceProvider();
```

```
        bufferPolicy = cryptoSHA1.ComputeHash(bufferPolicy);
        var providerRSA = new RSACryptoServiceProvider();
        providerRSA.FromXmlString(PRIVATE_KEY);
        var rsaFormatter = new RSAPKCS1SignatureFormatter(providerRSA);
        rsaFormatter.SetHashAlgorithm("SHA1");
        var signedPolicyHash = rsaFormatter.CreateSignature(bufferPolicy);
        var strSignedPolicy = System.Convert.ToBase64String(signedPolicyHash);

        // Build the query string with the expiration, policy signature,
        // and CloudFront key pair ID.
        var queryString = "Expires={0}&Signature={1}&Key-Pair-Id={2}";
        queryString = String.Format(queryString, Server.UrlEncode(expires),
Server.UrlEncode(strSignedPolicy), Server.UrlEncode(KEYPAIR_ID));
        var urlString = resourceUrl + "?" + queryString;
        return urlString;
    }

    /// <summary>
    /// Return a message saying this is blocked because of your location,
    /// or return an image tag.
    /// </summary>
    /// <returns></returns>
    public string GetContent()
    {
        var country = GetCountryCodeFromIP();
        if (COUNTRIES_TO_BLOCK.Contains(country))
        {
            // The country returned from the call to the geolocation service
            // is listed in the array of blocked countries.
            return string.Format(BLOCKED_MSG, country);
        }
        else
        {
            // The country returned from the call to the geolocation service
            // is NOT listed in the array of blocked countries
            // Get a signed URL for the content and display it.
            var url = GetSignedURL("CloudFront URL");
            var img = "<img src='{0}' />";
            return String.Format(img, url);
        }
    }
}
</script>
```

# Frequently Asked Questions

**How can I ensure that I retrieve the correct IP address of the end user visiting my website?**

You can use a variety of methods to get the IP address of the end user visiting your website. Here are two possible methods:

- If your web server is not connected to the Internet through a load balancer, you can use a web server variable to get the remote IP address. However, this IP address isn't always the end user's IP address—it can also be the IP address of a proxy server, depending on how the end user is connected to the Internet.

- If your web server is connected to the Internet through a load balancer, a web server variable may contain the IP address of the load balancer, not the IP address of the end user. In this configuration, we recommend that you use the last IP address in the `X-Forwarded-For` http header. This header

typically contains more than one IP address, most of which are for proxies or load-balancers. The last IP address in the list is the one most likely to be associated with the end user's geographic location.

If your web server is not connected to a load balancer, we recommend that you use web server variables instead of the `X-Forwarded-For` header in order to avoid IP address spoofing. The sample code in this document uses the `X-Forwarded-For` header if the header is present. If you do not want to use this method to get the IP address of the end user, you can edit the sample code.

**Can I use any third-party geolocation service to restrict access to my content in CloudFront?**

Yes. You will need an account with the third-party service to call their API, and you will need to modify the sample code accordingly.

**What is the cost of using this solution?**

The cost of using a third-party geolocation service will depend on which service provider you use. Current pricing for CloudFront usage is available on the Amazon CloudFront Pricing page. There are no additional CloudFront charges for using the CloudFront private-content feature.

**Can I use location information other than country to block access to my content?**

If your geolocation service provides information in addition to the country code, your application can use that information to determine whether you can distribute your content to the end user. Then your application can generate a CloudFront signed URL as described in this tutorial or in Using a Signed URL to Serve Private Content in the Amazon CloudFront Developer Guide.

**What should I do if the third-party service is not returning the correct information about an end user?**

Confirm that you are correctly calling the API provided by the third-party geolocation service, and that you are using the correct IP address for the end user. If you are still encountering issues with the third-party service or with the accuracy of the data that you receive from the service, contact the service vendor directly.

# Additional Services and Documentation

## Digital Element Services and Documentation

For information about Digital Element services, see the Digital Element website.

Documentation for Digital Element services is available only with a Digital Element account.

## MaxMind Services and Documentation

MaxMind offers a variety of geolocation services and other web services, including the following services:

- MaxMind GeoIP Omni Web Service, http://www.maxmind.com/app/web_services_omni
- MaxMind JavaScript Web Service, http://www.maxmind.com/app/javascript
- Other MaxMind web services, http://www.maxmind.com/app/web_services

The download distribution for each MaxMind API includes documentation and sample programs.

For more information, see Using MaxMind with Amazon CloudFront.

# Amazon Web Services Documentation

- CloudFront, http://aws.amazon.com/documentation/cloudfront
- Amazon S3, http://aws.amazon.com/documentation/s3/

# Amazon CloudFront Resources

The following table lists related resources that you'll find useful as you work with this service.

| Resource | Description |
|---|---|
| Amazon CloudFront Getting Started Guide | The getting started guide provides instructions for using the service for the first time. |
| Amazon CloudFront API Reference | The API reference gives the schema location; complete descriptions of the API actions, parameters, and data types; and a list of errors that the service returns. |
| Amazon CloudFront Release Notes | The release notes give a high-level overview of the current release. They specifically note any new features, corrections, and known issues. |
| Technical documentation for the Amazon Simple Storage Service (S3) | The technical documentation provides a detailed discussion of the service. It includes the basics of getting started, an overview of the service, programming reference, and API reference. |
| AWS Developer Tools | A central starting point to find documentation, code samples, release notes, and other information to help you build innovative applications with AWS. |
| AWS Management Console | The console allows you to perform most of the functions of Amazon CloudFront without programming. |
| Discussion Forums | A community-based forum for developers to discuss technical questions related to Amazon CloudFront. |
| AWS Support Center | The home page for AWS Technical Support, including access to our Developer Forums, Technical FAQs, Service Status page, and Premium Support (if you are subscribed to this program). |
| AWS Premium Support Information | The primary web page for information about AWS Premium Support, a one-on-one, fast-response support channel to help you build and run applications on AWS Infrastructure Services. |

| Resource | Description |
| --- | --- |
| Amazon CloudFront product information | The primary web page for information about Amazon CloudFront. |
| Contact Us | A central contact point for inquiries concerning AWS billing, account, events, abuse, etc. |
| Conditions of Use | Detailed information about the copyright and trademark usage at Amazon.com and other topics. |

# Document History

This documentation is associated with the 2012-03-15 release of Amazon CloudFront. This guide was last updated on 1 April 2012.

The following table describes the important changes since the last release of the *Amazon CloudFront Developer Guide*.

| Change | Description | Release Date |
|---|---|---|
| Updated Documentation | The documentation about working with objects was reorganized and clarified. For the revised documentation, see Working with Objects (p. 21). | 4 April 2012 |
| New Documentation | Documentation about live streaming with Microsoft IIS Media Services version 4.1 was added. For more information, see Live Smooth Streaming Using Amazon CloudFront and IIS Media Services 4.1 (p. 161). | 1 April 2012 |
| Updated Documentation | The documentation about live streaming with Adobe Flash Media Server was updated with information about Adobe Flash Media Server version 4.5. For more information, see Live HTTP Streaming Using CloudFront and Adobe Flash Media Server 4.5 (p. 139). | 29 March 2012 |
| New Feature | This release of CloudFront reduces the minimum TTL value for a download distribution. If you don't specify a minimum TTL when you create a distribution, CloudFront sets the minimum TTL to 0 seconds. For more information, go to the following documentation:<br><br>• CloudFront product page<br>• "Caching Duration and Minimum TTL" at Request and Response Behavior for Amazon S3 Origins (p. 63)<br>• "Caching Duration and Minimum TTL" at Request and Response Behavior, and Supported HTTP Status Codes for Custom Origins (p. 65)<br>• The `CachingBehavior` element in the DistributionConfig Complex Type. | 15 March 2012 |

| Change | Description | Release Date |
|--------|-------------|--------------|
| Updated Documentation | Topics about live streaming with Adobe Flash Media Server and about geoblocking were moved from a separate document into the CloudFront Tutorials (p. 139) chapter in this guide. | 2 February 2012 |
| New Feature | This release of Amazon CloudFront introduces AWS Management Console support for creating a distribution with a custom origin, restricting your distribution to HTTPS exclusively, and specifying a default root object. For more information, go to the Amazon CloudFront product page or see any of the following topics in the *Amazon CloudFront Developer Guide*: Creating a Distribution with a Custom Origin, Creating Secure HTTPS Connections, and Creating a Default Root Object. | 27 April 2011 |
| New Feature | This release of CloudFront includes integration with AWS Identity and Access Management (IAM). For more information, go to the Amazon CloudFront product page or Controlling User Access to Your AWS Account (p. 14) in the *Amazon CloudFront Developer Guide*. | 10 March, 2011 |
| New Feature | This release of CloudFront includes new APIs to support custom origins. For more information, go to the Amazon CloudFront product page or Creating a Distribution with a Custom Origin (p. 41) in the *Amazon CloudFront Developer Guide*. | 09 November 2010 |
| New Feature | This release of CloudFront includes new APIs for object invalidation. For more information, go to the Amazon CloudFront product page or Invalidating Objects (p. 26) in the *Amazon CloudFront Developer Guide*. | 31 August 2010 |
| New Feature | CloudFront now supports the ability to assign a default root object to your distribution. For more information, see Specifying a Default Root Object (p. 29). | 05 August 2010 |
| New Feature | Access logging for HTTP distributions now includes a field for query string parameters. For more information, see Download Distribution File Format (p. 123). | 14 July 2010 |
| New Feature | Added support for secure connections using HTTPS. For more information, see Creating Secure HTTPS Connections (p. 113). | 07 June 2010 |
| New Feature | Added logging for streaming content. For more information, see Streaming Distribution Log File Format (p. 124). | 13 May 2010 |
| New Feature | Reduced the minimum amount of time an object can be on an edge server from 24 hours to 1 hour. The default, however, remains 24 hours. For more information, see Expiration (p. 25). | 13 April 2010 |
| New Feature | Added feature to serve private streaming content over a Real-Time Messaging Protocol (RTMP) and prevent the content from being downloaded. For more information, see Overall Process to Serve Private Content (p. 73). | 28 March 2010 |
| New Feature | Added feature to deliver streaming content over a Real-Time Messaging Protocol (RTMP) connection. For more information, see Streaming Media on Demand (p. 56). | 15 December 2009 |

| Change | Description | Release Date |
|--------|-------------|--------------|
| New Feature | Added feature to restrict access to your content delivered over HTTP. For more information, see Using a Signed URL to Serve Private Content (p. 70). | 11 November 2009 |
| New Guide | We've separated the API reference material into its own guide. The *Amazon CloudFront Developer Guide* contains general information about how to use CloudFront, and the Auto Scaling API Reference contains detailed information about the control API requests, responses, and errors. | 11 November 2009 |