Amazon CloudFront

Developer Guide

API Version 2012-07-01

Amazon CloudFront: Developer Guide

Introduction to Amazon CloudFront

Topics

What Is Amazon CloudFront and Why Do I Need It?

CloudFront is a web service that speeds up distribution of your static and dynamic web content, for example, .html, .css, .php, and image files, to end users. CloudFront delivers your content through a worldwide network of edge locations. When an end user requests content that you're serving with CloudFront, the user is routed to the edge location that provides the lowest latency, so content is delivered with the best possible performance. If the content is already in that edge location, CloudFront delivers it immediately. If the content is not currently in that edge location, CloudFront retrieves it from an Amazon S3 bucket or an HTTP server (for example, a web server) that you have identified as the source for the definitive version of your content.

This concept is best illustrated by an example. Suppose you're serving the following image from a traditional web server, not from CloudFront:

(The image is owned by NASA and comes from the Visible Earth website, http://visibleearth.nasa.gov/.)

You're serving the image using the URL `http://example.com/globe_west_540.jpg`. Your end users can easily navigate to this URL and see the image, but they probably don't know that their request was routed from one network to another—through the complex collection of interconnected networks that comprise the Internet—until the image was found.

Further suppose that the web server from which you're serving the image is in Seattle, Washington, USA, and that an end user in Austin, Texas, USA requests the image. The traceroute list below (courtesy of www.WatchMouse.com) shows one way that this request could be routed.

1	vrid-225.core-sw.aus.us.siteprotect.com (216.139.225.1)	0.627 ms
2	xe-3-4.brdr-rtr-02.aus.us.siteprotect.com (216.139.253.53)	0.219 ms
3	66.113.197.121	0.452 ms
4	xe-5-2-0.edge3.Dallas1.Level3.net (4.59.112.37)	4.978 ms
5	ae-73-70.ebr3.Dallas1.Level3.net (4.69.145.116)	9.817 ms
6	ae-7-7.ebr3.Atlanta2.Level3.net (4.69.134.22)	30.570 ms
7	ae-2-2.ebr1.Washington1.Level3.net (4.69.132.86)	38.801 ms
8	ae-81-81.csw3.Washington1.Level3.net (4.69.134.138)	41.795 ms
9	ae-3-89.edge2.Washington1.Level3.net (4.68.17.145)	39.193 ms
10	72.21.222.139	35.767 ms

Map courtesy of the University of Texas Libraries, The University of Texas at Austin

In this example, the request was routed 10 times within the United States before the image was retrieved, which is not an unusually large number of hops. If your end user were in Europe, the request would be routed through even more networks to reach your server in Seattle. The number of networks and the distance that the request and the image must travel have a significant impact on the performance, reliability, and availability of the image.

CloudFront speeds up the distribution of your content by routing end users to the edge location that can best serve the end user's request in a worldwide network of edge locations; typically, requests are routed to the nearest CloudFront edge location in terms of latency. This dramatically reduces the number of networks that your users' requests must pass through and improves performance. End users get lower latency—the time it takes to load the first byte of the object—and higher data transfer rates. You also get increased reliability and availability because there is no longer a central point of failure. Copies of your object are now held in edge locations around the world.

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

How Do I...?

How Do I?	Relevant Topics
Get Started	Getting Started with CloudFront (p. 9)
Learn if CloudFront is right for my use case	Introduction to Amazon CloudFront (p. 1)
Use CloudFront distributions	Working with Distributions (p. 28)
Create a download distribution	Creating Download Distributions (p. 16)

How Do I?	Relevant Topics
Create a streaming distribution	Creating Streaming Distributions (p. 22)
Migrate from Amazon S3 to CloudFront	Migrating from Amazon S3 to CloudFront (p. 26)
Understand the security protocols used with CloudFront	Using an HTTPS Connection to Access Your Objects (p. 136)
Use the CloudFront API	Making API Requests (p. 157)

CloudFront Concepts

This section explains the basic concepts behind CloudFront.

Objects

Objects are the files you want CloudFront to deliver. Objects typically include web pages, images, and media files, but can be anything that can be served over HTTP or one of the versions of Adobe RTMP, the protocol used by Adobe Flash Media Server.

Origin Server

An origin server is the location where you store the original, definitive version of your objects. If you're serving content over HTTP, your origin server is either an Amazon S3 bucket or an HTTP server, for example, a web server. Your HTTP server can be running on an Amazon Elastic Compute Cloud instance or on a server that you manage; these servers are also known as *custom origins*.

If you're serving media files using RTMP, your origin server is always an Amazon S3 bucket.

Distributions

After you store your objects in your origin server, you create a distribution, which tells CloudFront where your objects are. When you create a distribution, CloudFront gives you a unique domain name that you use to reference your objects, for example, d111111abcdef8.cloudfront.net.

Suppose your origin is an Amazon S3 bucket called myawsbucket that contains an image called `/images/image.jpg`. You want to display the image on your web page. If you display the image directly from your Amazon S3 bucket, the URL is:

```
http://myawsbucket.s3.amazonaws.com/images/image.jpg
```

If you instead display the image using CloudFront, the URL is:

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

If you already have a domain name (for example, example.com) that you want to use for links to your objects, you can specify up to 10 alternate domain names, also known as CNAMEs. If you add the alternate domain name `www.example.com`, you can use the following URL for links to the image:

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

For more information about alternate domain names, see Using Alternate Domain Names (CNAMEs) (p. 52).

You can create two types of distributions:

- A **download distribution** delivers content using HTTP or HTTPS. Using a download distribution, you can configure CloudFront to access your web content in any combination of up to 10 Amazon S3 buckets and custom origins.
- A **streaming distribution** delivers digital media using Adobe Flash Media Server and the Real-Time Messaging Protocol. The origin for a streaming distribution is always one Amazon S3 bucket.

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 that edge location doesn't have the object, CloudFront gets a copy from the origin server, serves it to the end user, and stores it in the cache at that 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, the next time CloudFront gets a request for the object, the request is forwarded to your origin to determine whether an updated version is available. If you have updated the object on your origin server, the origin server returns the updated version. CloudFront serves it to the end user and stores the updated version in the cache at that edge location. The minimum expiration time is 0 seconds; there isn't a maximum expiration time limit. For more information about expiration, see Specifying How Long Objects Stay in a CloudFront Edge Cache (Object Expiration) (p. 65).

Eventual Consistency

When you create, modify, or delete a CloudFront distribution, it takes time for your changes to propagate to the CloudFront database. Information about the distribution is eventually consistent, but an immediate request for information about a distribution information might not show the change. Consistency is usually reached within minutes, but a high system load or network partition might increase this time.

How CloudFront Delivers Content

Here is an overview of how CloudFront delivers your content, including the steps you perform before your first end user accesses your application or website. Let's assume that you're using a download distribution to deliver image files from an Amazon S3 bucket and to serve HTML content from an HTTP server. The process for serving media files using a streaming distribution is very similar.

Let's also assume that you make the objects in your bucket publicly readable, so anyone who knows the CloudFront URLs for your objects can access them. If you want to keep the objects private and control who accesses them, see Serving Private Content through CloudFront (p. 87).

Process for Delivering Content

1. You configure your origin servers, which are the servers from which CloudFront gets your files for distribution at edge locations all over the world. For this example:

   - You create an Amazon S3 bucket for your image files, and you make the files publicly available.
   - You configure an HTTP server for your HTML content.

2. You upload your files to your origin servers.

3. You create a CloudFront distribution, which tells CloudFront which origin servers to get your files from when end users request the files through your web site or application. For this example, you configure your distribution so CloudFront gets your image files from one server and PHP files from another.

   As you create your distribution, you also specify details such as whether you want CloudFront to log all requests and whether you want the distribution to be enabled as soon as it's created.

4. CloudFront returns the domain name that you use for your objects as you develop your web site or application. For example, if CloudFront returns `d1234.cloudfront.net` as the domain name for your distribution, the URL for image.jpg in your Amazon S3 bucket (or in the root directory on an HTTP server) will be `http://d1234.cloudfront.net/image.jpg`.

   You can also configure your CloudFront distribution so you can use your own domain name. In that case, the URL might be `http://www.example.com/image.jpg`.

5. CloudFront distributes information about your distribution to all of its edge locations.

6. You develop your website or application to use the domain name that CloudFront returned in Step 4 or to use your own domain name, depending on how you configured your distribution.

   You can optionally configure your origin server to add headers to the files; the headers indicate how long you want the files to stay in the cache in CloudFront edge locations. By default, each object stays in an edge location for 24 hours.

7. An end user accesses your website or application and requests an image file and an HTML file.

8. CloudFront determines which edge location can best serve the end user's request, typically the nearest CloudFront edge location in terms of latency. In this example, let's say it's Tokyo.

9. In the Tokyo edge location, CloudFront checks its cache for the requested files. If the files are in the cache, CloudFront returns them to the end user. If the files are *not* in the cache:

   • CloudFront compares the request with the specifications in your distribution and forwards the request for the files to the applicable origin server for the corresponding file type: to your Amazon S3 bucket for image files and to your HTTP server for the HTML files.

   • The origin servers send the files back to the CloudFront edge location in Tokyo.

   • CloudFront forwards the files to the end user and adds the files to the cache in Tokyo for the next time someone requests those files.

10. After an object has been in an edge cache for 24 hours or for the duration specified in your file headers (see Step 6), CloudFront forwards the next request for the object to your origin to determine whether the edge location has the latest version. If the version in the edge location is the latest, CloudFront delivers it to your end user. If not, your origin sends the latest version to CloudFront, and CloudFront delivers the object to your end user and stores the latest version in the cache at that edge location. For more information, see Specifying How Long Objects Stay in a CloudFront Edge Cache (Object Expiration) (p. 65).

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. As with the other AWS services, you pay as you go and pay only for what you use.

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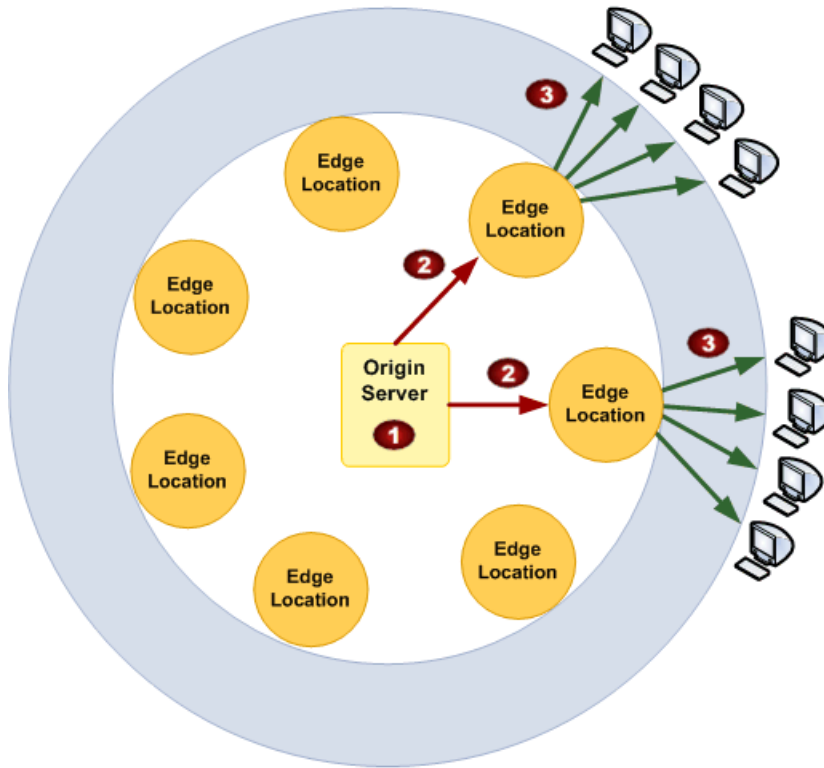
Your monthly bill from AWS separates your usage and dollar amounts by AWS service and function. As a result, you see some charges for storing objects with Amazon S3 (1) (if you are using Amazon S3 as your origin server), some charges for data transfer between your bucket and your edge location (2), and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

	Charge	Comments
3	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. For more information, see Amazon CloudFront Pricing.

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.

Getting Started with CloudFront

The example in this topic gives you a quick overview of how to use CloudFront to:

- Store the original versions of your objects in one Amazon Simple Storage Service (Amazon S3) bucket.
- Distribute download content such as text or graphics.
- Make your objects accessible to everyone.
- Use the CloudFront domain name in URLs for your objects (for example, `http://d111111abcdef8.cloudfront.net/image.jpg`) instead of your own domain name (for example, `http://www.example.com/image.jpg`).
- Keep your objects in CloudFront edge locations for the default duration of 24 hours. (The minimum duration is 0 seconds.)

For information about how to use CloudFront when you want to use other options, see Creating Download Distributions (p. 16) or Creating Streaming Distributions (p. 22).

You only need to perform a few basic steps to start delivering your content using CloudFront. The first step is signing up. After that, you create a CloudFront distribution, and then use the CloudFront domain name to reference content in your web pages or applications.

Step 1: Sign up for Amazon Web Services

If you haven't already done so, sign up for Amazon Web Services at http://aws.amazon.com. Just click **Sign Up Now** and enter any required information.

Step 2: Upload your content to Amazon S3 and grant object permissions

An Amazon S3 bucket is a container that can contain objects or folders. CloudFront can distribute almost any type of object for you using an Amazon S3 bucket as the source, for example, text, images, and videos. You can create multiple buckets, and there is no limit to the amount of data that you can store on Amazon S3.

By default, your Amazon S3 bucket and all of the objects in it are private—only the AWS account that created the bucket has permission to read or write the objects in it. If you want to allow anyone to access the objects in your Amazon S3 bucket using CloudFront URLs, you must grant public read permissions to the objects. (This is one of the most common mistakes when working with CloudFront and Amazon S3. You must explicitly grant privileges to each object in an Amazon S3 bucket.)

Note
If you want to restrict who can download your content, you can use the CloudFront private content feature. For more information about distributing private content, see Serving Private Content through CloudFront (p. 87).

To upload your content to Amazon S3 and grant read permission to everyone

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, click **Create Bucket**.

3. In the **Create Bucket** dialog, enter a bucket name.

   Important
   For your bucket to work with CloudFront, the name must conform to DNS naming requirements. For more information, go to Bucket Restrictions and Limitations in the *Amazon Simple Storage Service Developer Guide*.

4. Select a region for your bucket. By default, Amazon S3 creates buckets in the US-Standard region. We recommend that you choose a region close to you to optimize latency, minimize costs, or to address regulatory requirements.

5. Click **Create**.

6. Select your bucket in the **Buckets** pane, and click **Upload**.

7. On the **Upload - Select Files** page, click **Add Files**, and choose the files that you want to upload.
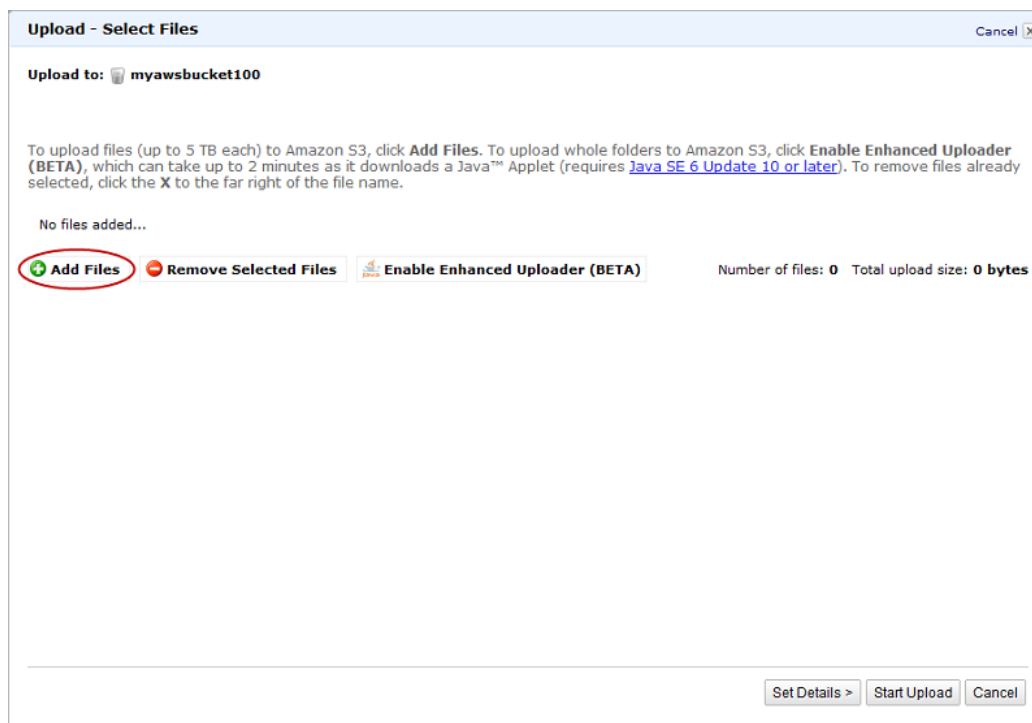
8. Enable public read privileges for each object that you upload to your Amazon S3 bucket.

   a. Click **Set Details**.

b. On the **Set Details** page, click **Set Permissions**.

c. On the **Set Permissions** page, click **Make everything public**.

9. Click **Start Upload**.

   After the upload completes, you can navigate to this item by its URL. In the case of the previous example, the URL would be:

   ```
   http://s3.amazonaws.com/example-myawsbucket/filename
   ```

   Use your Amazon S3 URL to verify that your content is publicly accessible, but remember that this is not the URL you will use when you are ready to distribute your content.

Step 3: Create a CloudFront Download Distribution

To create a CloudFront download distribution

1. Open the Amazon CloudFront console at https://console.aws.amazon.com/cloudfront/.
2. Click **Create Distribution**.
3. On the first page of the **Create Distribution Wizard**, accept the default selection, **Download**, and click **Continue**.
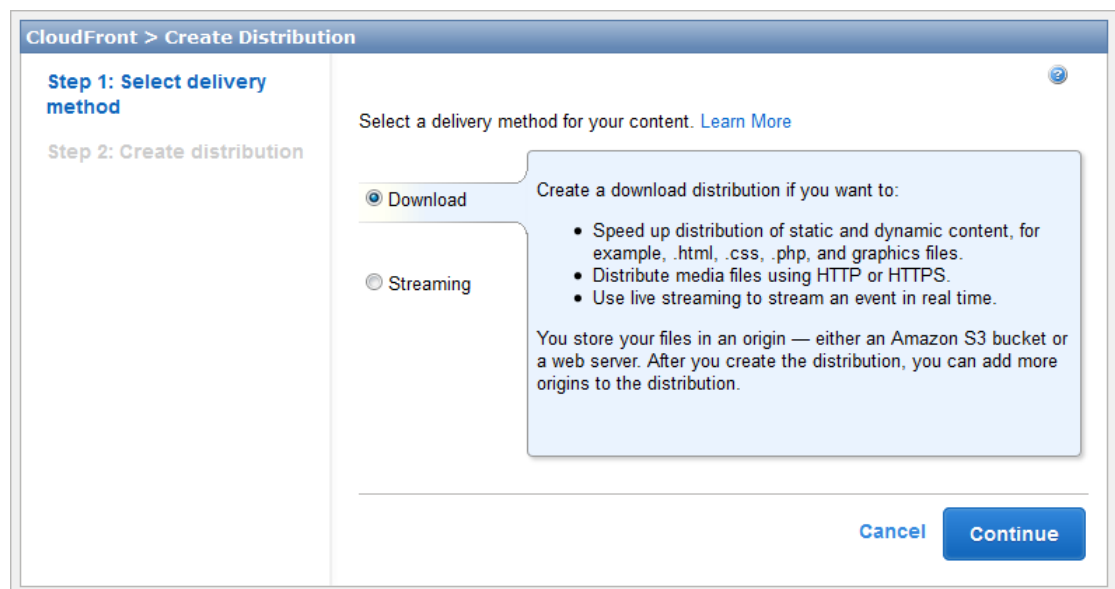
4. Select the Amazon S3 bucket that you created earlier. For **Origin ID** and **Restrict Bucket Access**, accept the default values.

5. Under **Default Cache Behavior Settings**, accept the default values, and CloudFront will:

    *   Forward all requests that use the CloudFront URL for your distribution (for example, `http://d111111abcdef8.cloudfront.net/image.jpg`) to the Amazon S3 bucket that you specified in Step 4.
    *   Allow end users to use either HTTP or HTTPS to access your objects.
    *   Cache your objects at CloudFront edge locations for 24 hours.
    *   Exclude cookies and query string parameters, if any, when forwarding requests for objects to your origin. (Amazon S3 doesn't process cookies and processes only a limited set of query string parameters.)
    *   Allow everyone to view your content.

    For more information about cache behavior options, see .

6. Under **Distribution Details**, enter the applicable values:

- **Price Class:** Select the price class that corresponds with the maximum price that you want to pay for CloudFront service. By default, CloudFront serves your objects from edge locations in all CloudFront regions.

  For more information about price classes and about how your choice of price class affects CloudFront performance for your distribution, go to Choosing the Price Class for a CloudFront Distribution (p. 55). For information about CloudFront pricing, including how price classes map to CloudFront regions, go to Amazon CloudFront Pricing.

- **Alternate Domain Names (CNAMEs):** Optional. Specify one or more domain names that you want to use for URLs for your objects instead of the domain name that CloudFront assigns when you create your distribution. For example, if you want the URL for the object:

  `/images/image.jpg`

  to look like this:

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

  instead of like this:

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

  you would create a CNAME for `www.example.com`. You can create up to 10 CNAMEs per distribution.

  > **Important**
  > If you add a CNAME for `www.example.com` to your distribution, you also need to create (or update) a CNAME record with your DNS service to route queries for `www.example.com` to `s5c39gqb8ow64r.cloudfront.net`. You must have permission to create a CNAME record with the DNS service provider for the domain. Typically, this means that you own the domain, but you may also be developing an application for the domain owner. For more information about CNAMEs, see Using Alternate Domain Names (CNAMEs) (p. 52).

- **Default Root Object:** Optional. The object that you want CloudFront to request from your origin (for example, `index.html`) when a viewer requests the root URL of your distribution (`http://www.example.com/`) instead of an object in your distribution (`http://www.example.com/product-description.html`). Specifying a default root object avoids exposing the contents of your distribution.

- **Logging:** Optional. If you want CloudFront to log information about each request for an object and store the log files in an Amazon S3 bucket, select **On**, and specify the bucket and an optional prefix for the names of the log files. There is no extra charge to enable logging, but you accrue the usual Amazon S3 charges for storing and accessing the files. CloudFront doesn't delete the logs automatically, but you can delete them at any time.

- **Cookie Logging:** In this example, we're using Amazon S3 as the origin for your objects, and Amazon S3 doesn't process cookies, so we recommend that you select **Off** for the value of **Cookie Logging**.

- **Comment:** Optional. Enter any comments that you want to save with the distribution.

- **Distribution State:** Select **Enabled** if you want CloudFront to begin processing requests as soon as the distribution is created, or select **Disabled** if you do not want CloudFront to begin processing requests after the distribution is created.

7. Click **Create Distribution**.

8. After CloudFront has created your distribution, the value of the **Status** column for your distribution will change from **InProgress** to **Deployed**. If you chose to enable the distribution, it will then be ready to process requests. This should take less than 15 minutes.

   The domain name that CloudFront assigns to your distribution appears in the list of distributions. (It also appears on the **General** tab for a selected distribution.)

Step 4: Test your links

After you've created your distribution, CloudFront knows where your Amazon S3 origin server is, and you know the domain name associated with the distribution. You can create a link to your Amazon S3 bucket content with that domain name, and have CloudFront serve it.

Note
You must wait until the status of your distribution changes to **Deployed** before testing your links.

To link to your objects

1. Copy the following HTML into a new file:

   • Replace <domain name> with the domain name that CloudFront assigned to your distribution.

- Replace <object name> with the name of a file in your Amazon S3 bucket.

```
<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 `d111111abcdef8.cloudfront.net` and your object was `image.jpg`, the URL for the link would be:

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

If your object is in a folder within your bucket, include the folder in the URL. For example, if image.jpg is located in an images folder, then the URL would be:

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

2. Save the text in a file that has a .html filename extension.
3. Open your web page in a browser to ensure that you can see your content. If you cannot see the content, confirm that you have performed all of the steps correctly. You can also see the tips in Troubleshooting (p. 153).

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

For more information on using CloudFront, go to Where Do I Go from Here? (p. 252).

Creating Download Distributions

This section describes how to create a CloudFront download distribution. For more information about download distributions, see Working with Download Distributions (p. 30).

Task List for Creating a Download Distribution

The following task list summarizes the process for creating a download distribution.

To Create a Download Distribution

1. Create one or more Amazon S3 buckets or configure HTTP servers as your origin servers. An origin is the location where you store the original version of your web content. When CloudFront gets a request for your files, it goes to the origin to get the files that it distributes at edge locations. You can use any combination of up to 10 Amazon S3 buckets and HTTP servers as your origin servers.

   If you're using Amazon S3, note that the name of your bucket must be all lowercase and cannot contain spaces.

   If you're using an Amazon EC2 server or another custom origin, review Requirements and Recommendations for Using Amazon EC2 and Other Custom Origins (p. 42).

2. Upload your content to your origin servers. If you don't want to restrict access to your content using CloudFront signed URLs, make the objects publicly readable.

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

3. Create your CloudFront download distribution:

   - For more information about creating a download distribution using the CloudFront console, see Creating a Download Distribution Using the CloudFront Console (p. 17).
   - For information about creating a download distribution using the CloudFront API, go to POST Distribution in the *Amazon CloudFront API Reference*.

4. Optional: If you created your distribution using the CloudFront console, create more cache behaviors or origins for your distribution. For more information, see To List, View, and Update CloudFront Distributions Using the CloudFront Console (p. 56).

5. Test your download distribution. For more information, see Testing Your Download Distribution (p. 20).

6. Develop your website or application to access your content using the domain name that CloudFront returned after you created your distribution in Step 3. For example, if CloudFront returns d111111abcdef8.cloudfront.net as the domain name for your distribution, the URL for the file `image.jpg` in an Amazon S3 bucket or in the root directory on an HTTP server will be `http://d111111abcdef8.cloudfront.net/image.jpg`.

   If you specified one or more alternate domain names (CNAMEs) when you created your distribution, you can use your own domain name. In that case, the URL for `image.jpg` might be `http://www.example.com/image.jpg`.

   Note the following:

   - If you want to use signed URLs to restrict access to your content, see Serving Private Content through CloudFront (p. 87).
   - If you want to serve compressed content, see Serving Compressed Files (p. 74).
   - For information about CloudFront request and response behavior for Amazon S3 and custom origins, see Request and Response Behavior, and Supported HTTP Status Codes (p. 78).

Creating a Download Distribution Using the CloudFront Console

The following procedure explains how to create a download distribution using the CloudFront console. If you want to create a download distribution using the CloudFront API, go to POST Distribution in the *Amazon CloudFront API Reference*.

By default, you can create a combined total of 100 download and streaming distributions for each AWS account. To request a higher limit, go to http://aws.amazon.com/cloudfront-request/.

To create a CloudFront download distribution using the CloudFront 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**.

3. On the first page of the **Create Distribution Wizard**, accept the default selection, **Download**, and click **Continue**.

4. Specify settings for the distribution. For more information, see Values that You Specify When You Create or Update a Download Distribution (p. 32).

Origin settings when the origin is an Amazon S3 bucket

Origin settings when the origin is an HTTP server

Default cache behavior settings

Distribution details

5. Click **Create Distribution**.

6. After CloudFront creates your distribution, the value of the **Status** column for your distribution will change from **InProgress** to **Deployed**. If you chose to enable the distribution, it will then be ready to process requests. This should take less than 15 minutes.

    The domain name that CloudFront assigns to your distribution appears in the list of distributions. (It also appears on the **General** tab for a selected distribution.)

When your distribution is deployed, confirm that you can access your content using your new CloudFront URL or CNAME. For more information, see .

Testing Your Download Distribution

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

Note

You must wait until the status of the distribution changes to **Deployed** before you can test your links.

To create links to objects in a download distribution

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

```
<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 were `d111111abcdef8.cloudfront.net` and your object were `image.jpg`, the URL for the link would be:

`http://d111111abcdef8.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 were located in the images folder on your origin server, then the URL would be:

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

2. Save the HTML code in a file that has a .html filename extension.
3. Open your web page in a browser to ensure that you can see your object.

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

Creating Streaming Distributions

This section describes how to configure on-demand streaming. If you're using Adobe Flash Player, Flowplayer, or JW Player for your media player, see the applicable tutorial instead:

- On-Demand Video Streaming Using CloudFront and Adobe Flash Player (p. 230)
- On-Demand Video Streaming Using CloudFront and Flowplayer for Adobe Flash (p. 237)
- On-Demand Video Streaming Using CloudFront and JW Player (p. 244)

Task List for Streaming Media Files

The following task list summarizes the process for creating a download distribution.

To Create a Streaming Distribution

1. Create an Amazon S3 bucket for your media files. If you are using a different Amazon S3 bucket for your media player, create an Amazon S3 bucket for the media player files, too.

   The names of your buckets must be all lowercase and cannot contain spaces.

2. Choose and configure a media player to play your media files. For more information, refer to the documentation for the media player.

3. Upload the files for your media player to the origin from which you want CloudFront to get the files. If you are using an Amazon S3 bucket as the origin for the media player, make the files (not the bucket) publicly readable.

4. Create a download distribution for your media player. (You can also use an existing distribution.) For more information, see *Creating Download Distributions* (p. 16).

5. 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.

   > **Important**
   > Media files in a Flash Video container must include the .flv filename extension, or the media will not stream.

   You can put media player files and media files in the same bucket.

6. Create a streaming distribution for your media files:

   - For more information about creating a download distribution using the CloudFront console, see Creating a Streaming Distribution Using the CloudFront Console (p. 23).

- For information about creating a download distribution using the CloudFront API, go to POST Streaming Distribution in the *Amazon CloudFront API Reference*.

7. Configure your media player. For more information, see Configuring the Media Player (p. 51).

If you have trouble getting your content to play, see Troubleshooting Streaming Distributions (p. 52).

Creating a Streaming Distribution Using the CloudFront Console

The following procedure explains how to create a streaming distribution using the CloudFront console. If you want to create a streaming distribution using the CloudFront API, go to POST Streaming Distribution in the *Amazon CloudFront API Reference*.

By default, you can create a combined total of 100 download and streaming distributions for each AWS account. To request a higher limit, go to http://aws.amazon.com/cloudfront-request/.

To create a streaming distribution using the CloudFront 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**.

3. On the first page of the **Create Distribution Wizard**, click **Streaming**, and click **Continue**.

4. Specify settings for the distribution. For more information, see Values that You Specify When You
   Create or Update a Streaming Distribution (p. 46).

Distribution settings, part 1

Distribution settings, part 2

5. Click **Create Distribution**.
6. After CloudFront creates your distribution, the value of the **Status** column for your distribution will change from **InProgress** to **Deployed**. If you chose to enable the distribution, it will then be ready to process requests. This should take less than 15 minutes.

   The domain name that CloudFront assigns to your distribution appears in the list of distributions. The domain name also appears on the **General** tab for a selected distribution.

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 by using the following process.

Process for Migrating from Amazon S3 to CloudFront

1. Create a CloudFront distribution using the process described in the applicable topic:

   - Creating Download Distributions (p. 16)
   - Creating Streaming Distributions (p. 22)

   When you create the distribution, specify the name of your Amazon S3 bucket as the origin server.

   > **Important**
   > For your bucket to work with CloudFront, the name must conform to DNS naming requirements. For more information, go to Bucket Restrictions and Limitations in the *Amazon Simple Storage Service Developer Guide.*

   If you're using a CNAME with Amazon S3, specify the CNAME for your distribution, too.

2. Create test links to publicly readable objects in your Amazon S3 bucket, and test the links. Use the domain name for the distribution that DNS name in the links, for example, `http://d111111abcdef8.cloudfront.net/images/image.jpg`.

   For more information about format of CloudFront URLs, see Format of URLs for CloudFront Objects (p. 58).

3. If you're using Amazon S3 CNAMEs, your application uses your domain name (for example, example.com) to reference the objects in your Amazon S3 bucket instead of using the name of your bucket (for example, myawsbucket.s3.amazonaws.com). To continue using your domain name to reference objects instead of using the CloudFront domain name for your distribution (for example, d111111abcdef8.cloudfront.net), you need to update your settings with your DNS service provider.

   For Amazon S3 CNAMEs to work, your DNS service provider has a CNAME resource record set for your domain that currently routes end-user queries for the domain to your Amazon S3 bucket. When someone requests the object:

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

the request is automatically rerouted, and the object they see is:

```
http://myawsbucket.s3.amazonaws.com/images/image.jpg
```

To route queries to your CloudFront distribution instead of your Amazon S3 bucket, you need to use the method provided by your DNS service provider to update the CNAME resource record set for your domain. This updated CNAME record will start to redirect DNS queries from your domain to the CloudFront domain name for your distribution. For more information, see the documentation provided by your DNS service provider.

> **Note**
> If you're using Route 53 as your DNS service, go to Creating, Changing, and Deleting Resource Record Sets for information about how to update a CNAME resource record set.

For more information about using CNAMEs with CloudFront, see Using Alternate Domain Names (CNAMEs) (p. 52).

After you update the CNAME resource record set, it can take up to 72 hours for the change to propagate throughout the DNS system, although it usually happens faster. During this time, some requests for your content will continue to be routed to your Amazon S3 bucket, and others will be routed to CloudFront.

Working with Distributions

Topics

A CloudFront distribution identifies the files that you want CloudFront to distribute through its edge locations and controls a variety of details about how you want CloudFront to distribute the files. When you create a distribution, you specify configuration settings such as:

- Where you want CloudFront to get the files that it distributes to edge locations—an Amazon S3 bucket or a web server.
- Whether you want the files to be available to everyone or you want to restrict access to selected users.
- Whether you want CloudFront to create access logs.

Changes to the CloudFront API

Beginning with the 2012-05-05 version of the CloudFront API, we made substantial changes to the format of the XML document that you include in the request body when you create or update a download distribution or a streaming distribution, and when you invalidate objects. With previous versions of the API, we discovered that it was too easy to accidentally delete one or more values for an element that accepts multiple values, for example, CNAMEs and trusted signers. Our changes for the 2012-05-05 release are intended to prevent these accidental deletions and to notify you when there's a mismatch between the number of values you say you're specifying in the $Quantity$ element and the number of values you're actually specifying.

Note the following about using the 2012-05-05 API version or later with download and streaming distributions that were created using earlier API versions:

- You cannot use versions of the API earlier than 2012-05-05 to update a download distribution that was created or updated using the 2012-05-05 or later CloudFront API.
- You can use the new API version to get a list of distributions, get information about a distribution, or get distribution configuration. CloudFront returns an XML document in the new XML format.
- To update a distribution that was created using an earlier API version, use the 2012-05-05 or later version of GET Distribution or GET Streaming Distribution to get an XML document in the new XML format, change the data as applicable, and use the 2012-05-05 or later version of PUT Distribution Config or PUT Streaming Distribution Config to submit the changes to CloudFront.
- You can use the new API to delete a distribution that was created using an earlier API version. The distribution must already be disabled.

Overview of Download and Streaming Distributions

> **Note**
> By default, you can create a combined total of 100 download and streaming distributions for each AWS account. To request a higher limit, go to http://aws.amazon.com/cloudfront-request/. The number of files that you can serve per distribution is unlimited.

Download distributions serve both static and dynamic content, for example, .html, .css, .php, and image files, using HTTP or HTTPS. A download distribution gets files from locations—known as origins—that you specify. Each origin is either an Amazon S3 bucket or an HTTP server, for example, a web server. You can specify any combination of up to 10 Amazon S3 buckets and/or HTTP servers as your origins.

For more information about how download distributions work, including the values that you specify when you create a download distribution, see Working with Download Distributions (p. 30). For information about creating a download distribution, see Creating Download Distributions (p. 16).

Streaming distributions stream media files using the Adobe Real-Time Messaging Protocol (RTMP). A streaming distribution must use an Amazon S3 bucket as the origin.

For information about the values you specify when you create a streaming distribution, see Working with Streaming Distributions (p. 43). For information about creating a streaming distribution, see Creating Streaming Distributions (p. 22).

Actions on Distributions

The following table lists the actions you can perform on a distribution and provides links to the corresponding documentation on how to perform the actions using the CloudFront console and the CloudFront API.

Action	Using the CloudFront Console	Using the CloudFront API: Download Distributions	Using the CloudFront API: Streaming Distributions
Create a distribution	**Download Distributions:** See Creating Download Distributions (p. 16) **Streaming Distributions:** See Creating Streaming Distributions (p. 22)	Go to POST Distribution	Go to POST Streaming Distribution
List your distributions	See Listing, Viewing, and Updating CloudFront Distributions (p. 55)	Go to GET Distribution List	Go to GET Streaming Distribution List
Get all information about a distribution	See Listing, Viewing, and Updating CloudFront Distributions (p. 55)	Go to GET Distribution	Go to GET Streaming Distribution
Get the distribution configuration	See Listing, Viewing, and Updating CloudFront Distributions (p. 55)	Go to GET Distribution Config	Go to GET Streaming Distribution Config
Update a distribution	See Listing, Viewing, and Updating CloudFront Distributions (p. 55)	Go to PUT Distribution Config	Go to PUT Streaming Distribution Config
Delete a distribution	See Deleting a Distribution (p. 56)	Go to DELETE Distribution	Go to DELETE Streaming Distribution

Working with Download Distributions

Topics
- Using Amazon S3 Origins and Custom Origins for Download Distributions (p. 31)
- Values that You Specify When You Create or Update a Download Distribution (p. 32)
- Values that CloudFront Displays in the Console When You Create or Update a Download Distribution (p. 41)
- Requirements and Recommendations for Using Amazon EC2 and Other Custom Origins (p. 42)

This section describes how you configure and manage CloudFront download distributions. For a basic explanation of distributions, see Working with Distributions (p. 28). For information about CloudFront streaming distributions, see Working with Streaming Distributions (p. 43).

Using Amazon S3 Origins and Custom Origins for Download Distributions

When you create a download distribution, you specify where CloudFront sends requests for the files that it distributes to edge locations. CloudFront supports using Amazon S3 buckets and HTTP servers (for example, web servers) as origins.

Using Amazon S3 Buckets for Your Origin

When you use Amazon S3 as an origin for your distribution, you place any objects that you want CloudFront to deliver in an Amazon S3 bucket. You can use any method that is supported by Amazon S3 to get your objects into Amazon S3, for example, the Amazon S3 console or API, or a third-party tool. You can create a hierarchy in your bucket to store the objects, just as you would with any other Amazon S3 bucket.

Using an existing Amazon S3 bucket as your CloudFront origin server doesn't change the bucket in any way; you can still use it as you normally would to store and access Amazon S3 objects at the standard Amazon S3 price. 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. 7).

> **Important**
> For your bucket to work with CloudFront, the name must conform to DNS naming requirements. For more information, go to Bucket Restrictions and Limitations in the *Amazon Simple Storage Service Developer Guide*.

When you specify the Amazon S3 bucket that you want CloudFront to get objects from, you use the following format:

bucket-name`.s3.amazonaws.com`

Do not specify the bucket using the following formats:

*   The Amazon S3 path style, `s3.amazonaws.com/`*bucket-name*
*   The Amazon S3 CNAME, if any

Using Amazon EC2 or Other Custom Origins

A custom origin is an HTTP server, for example, a web server. The HTTP server can be an Amazon EC2 instance or an HTTP server that you manage privately. When you use a custom origin, you specify the DNS name of the server, along with the HTTP and HTTPS ports and the protocol that you want CloudFront to use when fetching objects from your origin.

Most CloudFront features are supported when you use a custom origin with the following exceptions:

*   **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 more information, see Serving Private Content through CloudFront (p. 87).

For information about requirements and recommendations when using custom origins, see Requirements and Recommendations for Using Amazon EC2 and Other Custom Origins (p. 42).

Values that You Specify When You Create or Update a Download Distribution

When you create a new download distribution or update an existing distribution, you specify the following values.

For more information about creating or updating a download distribution, see the applicable topic:

- Creating a download distribution using the CloudFront console: Creating Download Distributions (p. 16)
- Updating a download distribution using the CloudFront console: Listing, Viewing, and Updating CloudFront Distributions (p. 55)
- Creating a download distribution using the CloudFront API: POST Distribution in the *Amazon CloudFront API Reference*
- Updating a download distribution using the CloudFront API: PUT Distribution Config in the *Amazon CloudFront API Reference*

Delivery Method

You specify the delivery method when you create a distribution. For a download distribution, this value is always **Download**. You can't change the delivery method for an existing distribution.

Origin Settings

When you create or update a distribution, you provide information about one or more locations—known as origins—where you store the original versions of your web content. CloudFront gets your web content from your origins and serves it to viewers via a world-wide network of edge servers. Each origin is either an Amazon S3 bucket or an HTTP server, for example, a web server.

By default, you can create a maximum of 10 origins for a distribution. To request a higher limit, go to http://aws.amazon.com/cloudfront-request/.

If you want to delete an origin, you must first edit or delete the cache behaviors that are associated with that origin.

> **Caution**
> If you delete an origin, confirm that files that were previously served by that origin are available in another origin and that your cache behaviors are now routing requests for those files to the new origin.

When you create or update a distribution, you specify the following values for each origin.

Origin Domain Name

The DNS domain name of the Amazon S3 bucket or HTTP server from which you want CloudFront to get objects for this origin, for example, `myawsbucket.s3.amazonaws.com` or `www.example.com`. In the CloudFront console, click in the **Origin Domain Name** field, and a list enumerates the Amazon S3 buckets that are associated with the current AWS account. To use a bucket from a different AWS account or to use a non-AWS resource, type the domain name of the resource. For example, for an Amazon S3 bucket, type the name in the following format:

`bucket-name``.s3.amazonaws.com`.

If the origin is an HTTP server, the files must be publicly readable.

If the origin is an Amazon S3 bucket, the files must be publicly readable unless you secure your content in Amazon S3 by using a CloudFront origin access identity. For more information, see Using an Origin Access Identity to Restrict Access to Your Amazon S3 Content (p. 92).

> **Important**
> If the origin is an Amazon S3 bucket, the bucket name must conform to DNS naming requirements. For more information, go to Bucket Restrictions and Limitations in the *Amazon Simple Storage Service Developer Guide*.

When you change the value of **Origin Domain Name** for an origin, CloudFront immediately begins replicating the change to CloudFront edge locations. Until the distribution configuration is updated in a given edge location, CloudFront will continue to forward requests to the previous HTTP server or Amazon

S3 bucket. As soon as the distribution configuration is updated in that edge location, CloudFront begins to forward requests to the new HTTP server or Amazon S3 bucket.

Changing the origin does not require CloudFront to repopulate edge caches with objects from the new origin. As long as the viewer requests in your application have not changed, CloudFront will continue to serve objects that are already in an edge cache until the TTL on each object expires or until seldom-requested objects are evicted.

Origin ID

A string that uniquely distinguishes this origin from other origins in this distribution. If you create cache behaviors in addition to the default cache behavior, you use the origin ID that you specify here to identify the origin to which you want CloudFront to route a request when the request matches the path pattern for that cache behavior. For more information, see Cache Behavior Settings (p. 35).

Restrict Bucket Access (Amazon S3 Only)

Click **Yes** if you want to require end users to access objects in an Amazon S3 bucket by using only CloudFront URLs, not by using Amazon S3 URLs. Then specify the applicable values.

Click **No** if you want end users to be able to access objects using either CloudFront URLs or Amazon S3 URLs.

For more information, see Using an Origin Access Identity to Restrict Access to Your Amazon S3 Content (p. 92).

Origin Access Identity (Amazon S3 Only)

If you chose **Yes** for **Restrict Bucket Access**, choose whether to create a new origin access identity or use an existing one that is associated with your AWS account. If you already have an origin access identity, we recommend that you reuse it to simplify maintenance. For more information about origin access identities, see Using an Origin Access Identity to Restrict Access to Your Amazon S3 Content (p. 92).

Comment for New Identity (Amazon S3 Only)

If you chose **Create a New Identity** for **Origin Access Identity**, enter a comment that identifies the new origin access identity. CloudFront will create the origin access identity when you create this distribution.

Your Identities (Amazon S3 Only)

If you chose **Use an Existing Identity** for **Origin Access Identity**, choose the origin access identity that you want to use. You cannot use an origin access identity that is associated with another AWS account.

Grant Read Permissions on Bucket (Amazon S3 Only)

If you want CloudFront to automatically grant the origin access identity the permission to read objects in your Amazon S3 bucket, click **Yes, Update Bucket Policy**.

> **Important**
> If you click **Yes, Update Bucket Policy**, CloudFront updates the bucket policy to grant the specified origin access identity the permission to read objects in your bucket. However, CloudFront does not remove existing permissions in the bucket policy or permissions on individual objects. If users currently have permission to access the objects in your bucket using Amazon S3 URLs, they will still have that permission after CloudFront updates your bucket policy. To view or change the existing bucket policy and the existing permissions on the objects in your bucket, use a method provided by Amazon S3. For more information, see Granting the Origin Access Identity Permission to Read Objects in Your Amazon S3 Bucket (p. 94).

If you want to update permissions manually, for example, if you want to update ACLs on your objects instead of updating bucket permissions, click **No, I will Update Permissions**.

Origin Protocol Policy (Amazon EC2 and Other Custom Origins Only)

The protocol policy that you want CloudFront to use when fetching objects from your origin server. If you specify **HTTP Only**, CloudFront only uses HTTP to access the origin.

If you specify **Match Viewer**, CloudFront fetches objects from your origin using HTTP or HTTPS, depending on the protocol of the viewer request. Note that CloudFront caches the object only once even if viewers make requests using both HTTP and HTTPS protocols.

HTTP Port (Amazon EC2 and Other Custom Origins Only)

Optional. The HTTP port that the custom origin listens on. Valid values include ports 80, 443, and 1024 to 65535. The default value is port 80.

HTTPS Port (Amazon EC2 and Other Custom Origins Only)

Optional. The HTTPS port that the custom origin listens on. Valid values include ports 80, 443, and 1024 to 65535. The default value is port 443.

Cache Behavior Settings

A cache behavior lets you configure a variety of CloudFront functionality for a given URL path pattern for files on your website. For example, one cache behavior might apply to all `.jpg` files in the `images` directory on a web server that you're using as an origin server for CloudFront. The functionality you can configure for each cache behavior includes:

- The path pattern.
- If you have configured multiple origins for your CloudFront distribution, which origin you want CloudFront to forward your requests to.
- Whether to forward query strings to your origin.
- Whether accessing the specified files requires signed URLs.
- Whether to require end users to use HTTPS to access those files.
- The minimum amount of time that those files stay in the CloudFront cache regardless of the value of any `Cache-Control` headers that your origin adds to the files.

When you create a new distribution, you specify settings for the default cache behavior, which automatically forwards all requests to the origin that you specify when you create the distribution. After you create a distribution, you can create additional cache behaviors that define how CloudFront responds when it receives a request for objects that match a path pattern, for example, `*.jpg`. If you create additional cache behaviors, the default cache behavior is always the last to be processed. Other cache behaviors are processed in the order in which they're listed in the CloudFront console or, if you're using the CloudFront API, the order in which they're listed in the `DistributionConfig` element for the distribution. For more information, see Path Pattern (p. 36).

When you create a cache behavior, you specify the one origin from which you want CloudFront to get objects. As a result, if you want CloudFront to distribute objects from all of your origins, you must have at least as many cache behaviors (including the default cache behavior) as you have origins. For example, if you have two origins and only the default cache behavior, the default cache behavior will cause CloudFront to get objects from one of the origins, but the other origin will never be used.

By default, you can create a maximum of 10 cache behaviors for a distribution. To request a higher limit, go to http://aws.amazon.com/cloudfront-request/.

Path Pattern

Note

When you're creating a new distribution, the value of **Path Pattern** defaults to * and cannot be changed. This value causes CloudFront to forward all requests for your objects to the origin that you specified in the Origin Domain Name (p. 33) field.

A path pattern (for example, `/images/*.jpg`) specifies which requests you want this cache behavior to apply to. When CloudFront receives an end-user request, the requested path is compared with path patterns in the order in which cache behaviors are listed in the distribution. The first match determines which cache behavior is applied to that request. For example, suppose you have three cache behaviors with the following three path patterns, in this order:

- `/images/*.jpg`
- `/images/*`
- `/*.gif`

A request for the file `/images/sample.gif` doesn't satisfy the first path pattern, so the associated cache behaviors are not be applied to the request. The file does satisfy the second path pattern, so the cache behaviors associated with the second path pattern are applied even though the request also matches the third path pattern.

Caution

Define path patterns and their sequence carefully or you may give end users undesired access to your content. For example, suppose a request matches the path pattern for two cache behaviors. The first cache behavior does not require signed URLs and the second cache behavior does require signed URLs. End users will be able to access the objects without using a signed URL because CloudFront processes the cache behavior associated with the first match.

The path you specify applies to requests for all files in the specified directory and in subdirectories below the specified directory. For example, if an `/images` directory contains `product1` and `product2` subdirectories, the path pattern `/images/*.jpg` applies to requests for any .jpg file in the `/images`, `/images/product1`, and `/images/product2` directories. If you want to apply a different cache behavior to the files in the `/images/product1` directory than the files in the `/images` and `/images/product2` directories, create a separate cache behavior for `/images/product1` and move that cache behavior to a position above (before) the cache behavior for the `/images` directory.

You can use the following wildcard characters in your path pattern:

- `*` matches 0 or more characters.
- `?` matches exactly 1 character.

The following examples show how the wildcard characters work:

Path pattern	Files that match the path pattern
`/*.jpg`	All .jpg files
`/images/*.jpg`	All .jpg files in the `/images` directory and in subdirectories under the `/images` directory
`/a*.jpg`	• All .jpg files for which the filename begins with `a`, for example, `apple.jpg` and `appalachian_trail_2012_05_21.jpg` • All .jpg files for which the file path begins with `a`, for example, `/abra/cadabra/magic.jpg`.

Path pattern	Files that match the path pattern
`/a??.jpg`	All .jpg files for which the filename begins with `a` and is followed by exactly two other characters, for example, `ant.jpg` and `abe.jpg`
`/*.doc*`	All files for which the filename extension begins with `.doc`, for example, `.doc`, `.docx`, and `.docm` files. You can't use the path pattern `*.doc?` in this case, because that path pattern wouldn't apply to requests for `.doc` files; the `?` wildcard character replaces exactly one character.

The path pattern for the default cache behavior is `*` (all files) and cannot be changed. If the request for an object does not match the path pattern for any of the other cache behaviors, CloudFront applies the behavior that you specify in the default cache behavior.

The maximum length of a path pattern is 255 characters. The value can contain any of the following characters:

- A-Z, a-z

- 0-9

- _ - . * $ / ~ " ' @ : +

- &, passed and returned as `&`

Origin (Existing Distributions Only)

When you're adding cache behaviors to an existing distribution or updating an existing origin, the value of **Origin ID** for the origin that you want CloudFront to route requests to when a request matches the path pattern either for a cache behavior or for the default cache behavior.

Viewer Protocol Policy

The protocol policy that you want viewers to use to access your content in the origin specified by **Origin**. If you specify **HTTP and HTTPS**, viewers can use both protocols. If you specify **HTTPS Only**, viewers are only allowed to access your content if they're using HTTPS. For more information, see Using an HTTPS Connection to Access Your Objects (p. 136).

Object Caching

If your origin server is adding a `Cache-Control` header to your objects to control how long the objects stay in the CloudFront cache, choose **Use Origin Cache Headers**.

To specify a minimum time that your objects stay in the CloudFront cache regardless of `Cache-Control` headers, choose **Customize**. Then, in the **Minimum TTL** field, specify the minimum number of seconds that you want objects to stay in the CloudFront cache even if `Cache-Control` headers specify a lower value.

Minimum TTL

The minimum amount of time that you want objects to stay in CloudFront caches before CloudFront queries your origin to see whether the object has been updated. For more information, see Specifying How Long Objects Stay in a CloudFront Edge Cache (Object Expiration) (p. 65).

Forward Cookies (Amazon EC2 and Other Custom Origins Only)

Specify whether you want CloudFront to forward cookies to your origin server and, if so, which ones. If you choose to forward only selected cookies (a whitelist of cookies), enter the cookie names in the **Whitelist Cookies** field. If you choose **All**, CloudFront forwards all cookies regardless of how many your application uses.

Amazon S3 doesn't process cookies, and forwarding cookies to the origin reduces cacheability. For cache behaviors that are forwarding requests to an Amazon S3 origin, choose **None** for **Forward Cookies**.

For more information about forwarding cookies to the origin, go to How CloudFront Forwards, Caches, and Logs Cookies (p. 62).

Whitelist Cookies (Amazon EC2 and Other Custom Origins Only)

If you chose **Whitelist** in the **Forward Cookies** list, in the **Whitelist Cookies** field, enter the names of cookies that you want CloudFront to forward to your origin server for this cache behavior. Enter each cookie name on a new line.

By default, if you choose the option to forward a whitelist of cookies to your origin, you can configure each cache behavior to forward a maximum of 10 cookie names to your origin server. To request a higher limit, go to http://aws.amazon.com/cloudfront-request/.

Forward Query Strings

If your origin server returns different versions of an object based on a query string in the URL, click **Yes**. If your origin returns the same version of an object regardless of the query string, click **No**. This increases the likelihood that CloudFront can serve a request from the cache, which improves performance and reduces the load on your origin. For more information about query strings, see How CloudFront Forwards, Caches, and Logs Query String Parameters (p. 61).

Restrict Viewer Access (Use Signed URLs)

If you want requests for objects that match the `PathPattern` for this cache behavior to use public URLs, click **No**.

If you want requests for objects that match the `PathPattern` for this cache behavior to use signed URLs, click **Yes**. Then specify the AWS accounts that you want to use to create signed URLs; these accounts are known as trusted signers.

For more information about trusted signers, see Specifying the AWS Accounts That Can Create Signed URLs (Trusted Signers) (p. 97).

Trusted Signers

Choose which AWS accounts you want to use as trusted signers for this cache behavior:

- **Self:** Use the account with which you're currently signed into the AWS Management Console as a trusted signer. If you're currently signed in as an IAM user, the associated AWS account is added as a trusted signer.
- **Specify Accounts:** Enter account numbers for trusted signers in the **AWS Account Numbers** field.

To create signed URLs, an AWS account must have at least one active CloudFront key pair.

> **Caution**
> If you're updating a distribution that you're already using to distribute content, add trusted signers only when you're ready to start generating signed URLs for your objects. After you add trusted

signers to a distribution, users must use signed URLs to access the objects that match the
`PathPattern` for this cache behavior.

AWS Account Numbers

If you want to create signed URLs using AWS accounts in addition to or instead of the current account,
enter one AWS account number per line in this field. Note the following:

- The accounts that you specify must have at least one active CloudFront key pair. For more information,
  see Creating CloudFront Key Pairs for Your Trusted Signers (p. 98).
- You can't create CloudFront key pairs for IAM users, so you can't use IAM users as trusted signers.
- To get the AWS account number for an account, sign into the AWS Management Console using that
  account, and click **My Account/Console > Security Credentials**. The account number for the current
  account appears at the top of the page.
- If you enter the account number for the current account, CloudFront automatically checks the **Self**
  checkbox and removes the account number from the **AWS Account Numbers** list.

Distribution Details

The following values apply to the entire distribution.

Price Class

The price class that corresponds with the maximum price that you want to pay for CloudFront service.
By default, CloudFront serves your objects from edge locations in all CloudFront regions.

For more information about price classes and about how your choice of price class affects CloudFront
performance for your distribution, see Choosing the Price Class for a CloudFront Distribution (p. 55). For
information about CloudFront pricing, including how price classes map to CloudFront regions, go to
Amazon CloudFront Pricing.

Alternate Domain Names (CNAMEs)

Optional. Specify one or more domain names that you want to use for URLs for your objects instead of
the domain name that CloudFront assigns when you create your distribution. For example, if you want
the URL for the object:

`/images/image.jpg`

to look like this:

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

instead of like this:

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

add a CNAME for `www.example.com`.

> **Important**
> If you add a CNAME for `www.example.com` to your distribution, you also need to create (or
> update) a CNAME record with your DNS service to route queries for `www.example.com` to
> `d111111abcdef8.cloudfront.net`. You must have permission to create a CNAME record
> with the DNS service provider for the domain. Typically, this means that you own the domain,
> but you may also be developing an application for the domain owner.

By default, you can create a maximum of 10 CNAMEs per distribution. To request a higher limit, go to
http://aws.amazon.com/cloudfront-request/.

For more information about alternate domain names, see Using Alternate Domain Names (CNAMEs) (p. 52). For more information about CloudFront URLs, see Format of URLs for CloudFront Objects (p. 58).

Default Root Object

Optional. The object that you want CloudFront to request from your origin (for example, `index.html`) when a viewer requests the root URL of your distribution (`http://www.example.com/`) instead of an object in your distribution (`http://www.example.com/product-description.html`). Specifying a default root object avoids exposing the contents of your distribution.

The maximum length of the name is 255 characters. The name can contain any of the following characters:

- A-Z, a-z
- 0-9
- _ - . * $ / ~ " '
- &, passed and returned as `&`

For more information, see Specifying a Default Root Object (Download Distributions Only) (p. 72).

Logging

Whether you want CloudFront to log information about each request for an object and store the log files in an Amazon S3 bucket. You can enable or disable logging at any time. There is no extra charge if you enable logging, but you accrue the usual Amazon S3 charges for storing and accessing the files in an Amazon S3 bucket. You can delete the logs at any time. For more information about CloudFront access logs, see Access Logs (p. 143).

Bucket for Logs

If you chose **On** for **Logging**, the Amazon S3 bucket that you want CloudFront to store access logs in, for example, `myawslogbucket.s3.amazonaws.com`. If you enable logging, CloudFront records information about each end-user request for an object and stores the files in the specified Amazon S3 bucket. You can enable or disable logging at any time. For more information about CloudFront access logs, see Access Logs (p. 143).

Log Prefix

Optional. If you chose **On** for **Logging**, specify the string, if any, that you want CloudFront to prefix to the access log filenames for this distribution, for example, `exampleprefix/`. The trailing slash (`/`) is optional but recommended to simplify browsing your log files. For more information about CloudFront access logs, see Access Logs (p. 143).

Cookie Logging

If you want CloudFront to include cookies in access logs, choose **On**. If you choose to include cookies in logs, CloudFront logs all cookies regardless of how you configure the cache behaviors for this distribution: forward all cookies, forward no cookies, or forward a specified list of cookies to the origin.

Amazon S3 doesn't process cookies, so unless your distribution also includes an Amazon EC2 or other custom origin, we recommend that you choose **Off** for the value of **Cookie Logging**.

For more information about cookies, go to How CloudFront Forwards, Caches, and Logs Cookies (p. 62).

Amazon CloudFront Developer Guide
Values that CloudFront Displays in the Console When
You Create or Update a Download Distribution

Comment

Optional. When you create a distribution, you can include a comment of up to 128 characters. You can update the comment at any time.

Distribution State

Indicates whether you want the distribution to be enabled or disabled once it's deployed:

- *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. 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. 5).
- *Disabled* means that even though the distribution might be deployed and ready to use, end users can't use it. 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 want. Follow the process for updating a distribution's configuration. For more information, see Listing, Viewing, and Updating CloudFront Distributions (p. 55).

Values that CloudFront Displays in the Console When You Create or Update a Download Distribution

When you create a new download distribution or update an existing distribution, CloudFront displays the following information in the CloudFront console.

> **Note**
> Active trusted signers, the AWS accounts that have an active CloudFront key pair and can be used to create valid signed URLs, are currently not visible in the CloudFront console.

Distribution ID (General Tab)

When you perform an action on a distribution using the CloudFront API, you use the distribution ID to specify which distribution you want to perform the action on, for example, EDFDVBD6EXAMPLE. You can't change the distribution ID.

Distribution Status (General Tab)

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
In addition to ensuring that the status for a distribution is **Deployed**, you must enable the distribution before end users can use CloudFront to access your content. For more information, see Distribution State (p. 41).

Last Modified (General Tab)

The date and time that the distribution was last modified, using ISO 8601 format, for example, 2012-05-19T19:37:58Z. For more information, go to http://www.w3.org/TR/NOTE-datetime.

Domain Name (General Tab)

You use the distribution's domain name in the links to your objects. For example, if your distribution's domain name is `d111111abcdef8.cloudfront.net`, the link to `/images/image.jpg` would be `http://d111111abcdef8.cloudfront.net/images/image.jpg`. You can't change the CloudFront domain name for your distribution. For more information about CloudFront URLs for links to your objects, see Format of URLs for CloudFront Objects (p. 58).

If you specified one or more alternate domain names (CNAMES), you can use your own domain names for links to your objects instead of using the CloudFront domain name. For more information about CNAMES, see Alternate Domain Names (CNAMEs) (p. 39).

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.

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. 78).

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

1. Use an Amazon Machine Image that automatically installs the software for a web server. For more information, see the Amazon EC2 documentation.
2. 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.
3. When you create your CloudFront distribution, specify the URL of the load balancer for the domain name of your origin server. For more information, see Creating Download Distributions (p. 16).

Working with Streaming Distributions

Topics

This section describes how you configure and manage streaming distributions. For information about how to create a streaming distribution, see Creating Streaming Distributions (p. 22).

How Streaming Distributions Work

To stream media files using CloudFront, you provide two types of files to your end users:

- Your media files
- A media player, for example, JW Player, Flowplayer, or Adobe Flash

End users view your media files using the media player that you provide for them; they do not use the media player (if any) that is already installed on their computer or other device.

When an end user streams your media file, the media player begins to play the content of the file while the file is still being downloaded from CloudFront. The media file is not stored locally on the end user's system.

To use CloudFront to serve both the media player and the media files, you need 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, while streaming distributions stream media files over RTMP (or a variant of RTMP).

The following example assumes that your media files and your media player are stored in different buckets in Amazon S3, but that isn't required—you can store media files and your media player in the same Amazon S3 bucket. You can also make the media player available to end users in other ways, for example, using CloudFront and a custom origin. However, the media files must use an Amazon S3 bucket as the origin.

In the following diagram, 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.
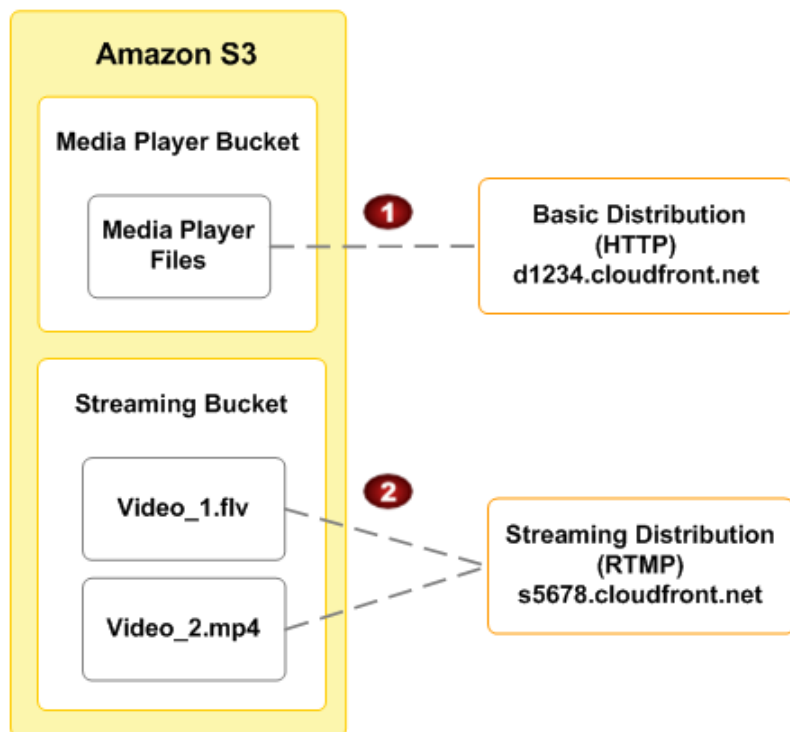
1	Your media player bucket holds the media player and is the origin server for a regular HTTP distribution. In this example, the domain name for the distribution is `d1234.cloudfront.net`. (The `d` in `d1234.cloudfront.net` indicates that this is a download distribution.)
2	Your streaming media bucket holds your media files and is the origin server for a streaming distribution. In this example, the domain name for the distribution is `s5678.cloudfront.net`. (The `s` in `s5678.cloudfront.net` indicates that this is a streaming distribution.)

When you configure CloudFront to distribute media files, CloudFront uses Adobe Flash Media Server 3.5 as the streaming server and streams your media files using 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

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. The overview includes information about supported codecs and containers.

Resources available on the Internet can help you determine the bit rate to use for your Flash files, for example, the Flash video (FLV) bitrate calculator on the Adobe website.

CloudFront supports all of the features in Adobe Flash Media Server 3.5 related to *dynamic streaming*, which lets you switch between streams of different qualities 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/. Alternatively, you can use an existing player. For more infomation, see the following tutorials:

- On-Demand Video Streaming Using CloudFront and Adobe Flash Player (p. 230)
- On-Demand Video Streaming Using CloudFront and Flowplayer for Adobe Flash (p. 237)
- On-Demand Video Streaming Using CloudFront and JW Player (p. 244)

Using an Amazon S3 Bucket as the Origin for a Streaming Distribution

When you create a distribution, you specify where CloudFront gets the files that it distributes to edge locations. For a streaming distribution, you must use an Amazon S3 bucket; custom origins are not supported. To get your objects into your bucket, you can use any method supported by Amazon S3, for example, the Amazon S3 API or a third-party tool. You can create a hierarchy in your bucket just as 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. 7).

Using an existing Amazon S3 bucket as your CloudFront origin server doesn't change the 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 can use the same Amazon S3 bucket for both streaming and download distributions.

> **Note**
> After you create a streaming distribution, you can't change its origin server. If you need to change the Amazon S3 bucket for a streaming distribution, you must create a new distribution that uses the new bucket and update either your links or your DNS records to use the domain name for the new distribution. You can then delete the original distribution. For more information, see Deleting a Distribution (p. 56).

When you specify the name of the Amazon S3 bucket that you want CloudFront to get objects from, you use the following format:

`bucket-name.s3.amazonaws.com`

Do not specify the name of the bucket using the following values:

- The Amazon S3 path style, `s3.amazonaws.com/bucket-name`
- The Amazon S3 CNAME, if any

> **Important**
> For your bucket to work with CloudFront, the name must conform to DNS naming requirements. For more information, go to Bucket Restrictions and Limitations in the *Amazon Simple Storage Service Developer Guide*.

Creating Multiple Streaming Distributions for an Origin Server

You typically create one streaming distribution per Amazon S3 bucket, but you can choose to create multiple streaming distributions for the same bucket. For example, if you had two distributions for an Amazon S3 bucket, you could reference a single media file using either distribution. In this case, if you had a media file called `media.flv` in your origin server, CloudFront would work with each distribution

as though it referenced an individual `media.flv` object: one `media.flv` accessible through one distribution, and another `media.flv` accessible through the other distribution.

Values that You Specify When You Create or Update a Streaming Distribution

To stream media files using CloudFront, you create a streaming distribution and specify the following values.

Topics

Origin Domain Name (Amazon S3 Bucket)

The DNS domain name of the Amazon S3 bucket from which you want CloudFront to get objects for this origin, for example, `myawsbucket.s3.amazonaws.com`. In the CloudFront console, click in the **Origin Domain Name** field, and a list enumerates the Amazon S3 buckets that are associated with the current AWS account. To use a bucket from a different AWS account, type the domain name of the bucket in the following format:

bucket-name`.s3.amazonaws.com.`

The files must be publicly readable unless you secure your content in Amazon S3 by using a CloudFront origin access identity. For more information, see Using an Origin Access Identity to Restrict Access to Your Amazon S3 Content (p. 92).

> **Important**
> The bucket name must conform to DNS naming requirements. For more information, go to Bucket Restrictions and Limitations in the *Amazon Simple Storage Service Developer Guide*.

When you change the bucket from which CloudFront gets objects for the current origin, CloudFront immediately begins replicating the change to CloudFront edge locations. Until the distribution configuration is updated in a given edge location, CloudFront will continue to forward requests to the previous Amazon S3 bucket. As soon as the distribution configuration is updated in that edge location, CloudFront begins to forward requests to the new Amazon S3 bucket.

Changing the bucket does not require CloudFront to repopulate edge caches with objects from the new origin. As long as the viewer requests in your application have not changed, CloudFront will continue to serve objects that are already in an edge cache until the TTL on each object expires or until seldom-requested objects are evicted.

For more information, see Using an Amazon S3 Bucket as the Origin for a Streaming Distribution (p. 45).

Restrict Bucket Access (Amazon S3 Only)

Click **Yes** if you want to require end users to access objects in an Amazon S3 bucket by using only CloudFront URLs, not by using Amazon S3 URLs. Then specify the applicable values.

Click **No** if you want end users to be able to access objects using either CloudFront URLs or Amazon S3 URLs.

For more information, see Using an Origin Access Identity to Restrict Access to Your Amazon S3 Content (p. 92).

Origin Access Identity (Amazon S3 Only)

If you chose **Yes** for **Restrict Bucket Access**, choose whether to create a new origin access identity or use an existing one that is associated with your AWS account. If you already have an origin access identity, we recommend that you reuse it to simplify maintenance. For more information about origin access identities, see Using an Origin Access Identity to Restrict Access to Your Amazon S3 Content (p. 92).

Comment for New Identity(Amazon S3 Only)

If you chose **Create a New Identity** for **Origin Access Identity**, enter a comment that identifies the new origin access identity. CloudFront will create the origin access identity when you create this distribution.

Your Identities (Amazon S3 Only)

If you chose **Use an Existing Identity** for **Origin Access Identity**, choose the origin access identity that you want to use. You cannot use an origin access identity that is associated with another AWS account.

Grant Read Permissions on Bucket (Amazon S3 Only)

If you want CloudFront to automatically grant the origin access identity the permission to read objects in your Amazon S3 bucket, click **Yes, Update Bucket Policy**.

> **Important**
> If you click **Yes, Update Bucket Policy**, CloudFront updates the bucket policy to grant the specified origin access identity the permission to read objects in your bucket. However, CloudFront does not remove existing permissions in the bucket policy or permissions on individual objects. If users currently have permission to access the objects in your bucket using Amazon S3 URLs, they will still have that permission after CloudFront updates your bucket policy. To view or change the existing bucket policy and the existing permissions on the objects in your bucket, use a method provided by Amazon S3. For more information, see Granting the Origin Access Identity Permission to Read Objects in Your Amazon S3 Bucket (p. 94).

If you want to update permissions manually, for example, if you want to update ACLs on your objects instead of updating bucket permissions, click **No, I will Update Permissions**.

Price Class

The price class that corresponds with the maximum price that you want to pay for CloudFront service. By default, CloudFront serves your objects from edge locations in all CloudFront regions.

For more information about price classes and about how your choice of price class affects CloudFront performance for your distribution, see Choosing the Price Class for a CloudFront Distribution (p. 55). For information about CloudFront pricing, including how price classes map to CloudFront regions, go to Amazon CloudFront Pricing.

Alternate Domain Names (CNAMEs)

Optional. You can associate one or more CNAME aliases with a distribution so that you can use your domain name (for example, example.com) in the URLs for your objects instead of using the domain name that CloudFront assigned when you created your distribution. For more information, see Using Alternate Domain Names (CNAMEs) (p. 52).

Logging

Whether you want CloudFront to log information about each request for an object and store the log files in an Amazon S3 bucket. You can enable or disable logging at any time. There is no extra charge if you enable logging, but you accrue the usual Amazon S3 charges for storing and accessing the files in an Amazon S3 bucket. You can delete the logs at any time. For more information about CloudFront access logs, see Access Logs (p. 143).

Bucket for Logs

If you chose **On** for **Logging**, the Amazon S3 bucket that you want CloudFront to store access logs in, for example, `myawslogbucket.s3.amazonaws.com`. If you enable logging, CloudFront records information about each end-user request for an object and stores the files in the specified Amazon S3 bucket. You can enable or disable logging at any time. For more information about CloudFront access logs, see Access Logs (p. 143).

Log Prefix

Optional. If you chose **On** for **Logging**, specify the string, if any, that you want CloudFront to prefix to the access log filenames for this distribution, for example, `exampleprefix/`. The trailing slash (/) is optional but recommended to simplify browsing your log files. For more information about CloudFront access logs, see Access Logs (p. 143).

Comment

Optional. When you create a distribution, you can include a comment of up to 128 characters. You can update the comment at any time.

Distribution State

When you create a distribution, you must specify whether you want the distribution to be enabled or disabled after 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. Whenever a distribution is enabled, CloudFront accepts and processes 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. 5).

- *Disabled* means that even though the distribution might be deployed and ready to use, end users can't use it. When 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 want. For information about updating a distribution's configuration, see Listing, Viewing, and Updating CloudFront Distributions (p. 55).

Restrict Viewer Access (Use Signed URLs)

If you want requests for objects served by this distribution to use public URLs, click **No**. If you want requests to use signed URLs, click **Yes**. Then specify the AWS accounts that you want to use to create signed URLs; these accounts are known as trusted signers.

For more information about trusted signers, see Specifying the AWS Accounts That Can Create Signed URLs (Trusted Signers) (p. 97).

Trusted Signers

Choose which AWS accounts you want to use as trusted signers for this distribution:

- **Self:** Use the account with which you're currently signed into the AWS Management Console as a trusted signer. If you're currently signed in as an IAM user, the associated AWS account is added as a trusted signer.
- **Specify Accounts:** Enter account numbers for trusted signers in the **AWS Account Numbers** field.

To create signed URLs, an AWS account must have at least one active CloudFront key pair.

> **Caution**
> If you're updating a distribution that you're already using to distribute content, add trusted signers only when you're ready to start generating signed URLs for your objects. After you add trusted signers to a distribution, users must use signed URLs to access the objects served by this distribution.

AWS Account Numbers

If you want to create signed URLs using AWS accounts in addition to or instead of the current account, enter one AWS account number per line in this field. Note the following:

- The accounts that you specify must have at least one active CloudFront key pair. For more information, see Creating CloudFront Key Pairs for Your Trusted Signers (p. 98).
- You can't create CloudFront key pairs for IAM users, so you can't use IAM users as trusted signers.
- To get the AWS account number for an account, sign into the AWS Management Console using that account, and click **My Account/Console > Security Credentials**. The account number for the current account appears at the top of the page.
- If you enter the account number for the current account, CloudFront automatically checks the **Self** checkbox and removes the account number from the **AWS Account Numbers** list.

Amazon CloudFront Developer Guide
Values that CloudFront Displays in the Console When
You Create or Update a Streaming Distribution

Values that CloudFront Displays in the Console When You Create or Update a Streaming Distribution

When you create a new streaming distribution or update an existing distribution, CloudFront displays the following information in the CloudFront console.

> **Note**
> Active trusted signers, the AWS accounts that have an active CloudFront key pair and can be used to create valid signed URLs, are currently not visible in the CloudFront console.

Distribution ID

When you perform an action on a distribution using the CloudFront API, you use the distribution ID to specify which distribution you want to perform the action on, for example, EDFDVBD6EXAMPLE. You can't change the distribution ID.

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.

In addition to ensuring that the status for a distribution is **Deployed**, you must enable the distribution before end users can use CloudFront to access your content. For more information, see Distribution State (p. 48).

Last Modified

The date and time that the distribution was last modified, using ISO 8601 format, for example, 2012-05-19T19:37:58Z. For more information, go to http://www.w3.org/TR/NOTE-datetime.

Domain Name

You use the distribution's domain name in the links to your objects, unless you're using alternate domain names (CNAMEs). 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. You can't change the CloudFront domain name for your distribution. For more information about CloudFront URLs for links to your objects, see Format of URLs for CloudFront Objects (p. 58).

If you specified one or more alternate domain names (CNAMEs), you can use your own domain names for links to your objects instead of using the CloudFront domain name. For more information about CNAMEs, see Alternate Domain Names (CNAMEs) (p. 39).

> **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.

Configuring the Media Player

To play a media file, you must 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 example above 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 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 Amazon S3 bucket, CloudFront ignores it.

Error Codes for Streaming Distributions

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

Error	Description
DistributionNotFound	The distribution was not found.

Error	Description
DistributionTypeMismatch	The distribution is not a streaming distribution.
InvalidInstance	The instance is invalid.
InvalidURI	The URI is invalid.

Troubleshooting Streaming Distributions

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

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), and 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	Confirm that the path for the file includes /cfx/st. You don't need to include /cfx/st in the path to the object in the Amazon S3 bucket. For more information, see Configuring the Media Player (p. 51).
MPEG-4 file names	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. 51).
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. To instead have the player send a message if the video is missing, use play("vid",0,-1) instead of play("vid").

Using Alternate Domain Names (CNAMEs)

In CloudFront, an alternate domain name, also known as a CNAME, lets you use your own domain name (for example, www.example.com) for links to your objects instead of using the domain name that CloudFront assigns to your distribution. Both download and streaming distributions support alternate domain names.

When you create a distribution, CloudFront returns a domain name for the distribution, for example:

d111111abcdef8.cloudfront.net

If you were to use the CloudFront domain name for your objects, the URL for an object called /images/image.jpg would be:

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

If you want to use your own domain name, `www.example.com`, for the URLs for your objects instead of the `cloudfront.net` domain name that CloudFront assigned to your distribution, you can add an alternate domain name to your distribution for `www.example.com`. You can then use the following URL for `/images/image.jpg`:

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

> **Important**
> Note the following restrictions on using CNAMEs:
>
> - By default, you can add a maximum of 10 alternate domain names per distribution. To request a higher limit, go to http://aws.amazon.com/cloudfront-request/.
> - You must have permission to create a CNAME record with the DNS service provider for the domain. Typically, this means that you own the domain, but you may also be developing an application for the domain owner.
> - You cannot add an alternate domain name to a CloudFront distribution if the alternate domain name already exists in another CloudFront distribution, even if your AWS account owns the other distribution.
> - The DNS protocol does not allow you to create a CNAME record for the top node of a DNS namespace, also known as the zone apex. For example, if you register the DNS name `example.com`, the zone apex is `example.com`. You cannot create a CNAME record for `example.com`, but you can create CNAME records for `www.example.com`, `newproduct.example.com`, and so on.
> - CloudFront doesn't support alternate domain names with HTTPS. For more information, see Alternate Domain Names (CNAMEs) and HTTPS (p. 138).

Adding an Alternate Domain Name

The following task list describes the process for using the CloudFront console to add an alternate domain name to your distribution so you can use your own domain name in your links instead of the CloudFront domain name that is associated with your distribution.

You can also update your distribution using the CloudFront API:

- To update a download distribution, use the `PUT Distribution Config` API action. For more information, go to PUT Distribution Config in the *Amazon CloudFront API Reference*.
- To update a streaming distribution, use the `PUT Streaming Distribution Config` API action. For more information, go to PUT Streaming Distribution Config in the *Amazon CloudFront API Reference*.

Process for Adding an Alternate Domain Name Using the CloudFront Console

1. Sign in to the AWS Management Console and open the Amazon CloudFront console at https://console.aws.amazon.com/cloudfront/.
2. In the CloudFront console, update your distribution to include your domain name as an alternate domain name in the **Alternate Domain Names (CNAMEs)** field. For more information, see Listing, Viewing, and Updating CloudFront Distributions (p. 55).
3. In the CloudFront console, on the General tab for your distribution, confirm that the status of your distribution has changed to **Deployed**. If you try to use an alternate domain name before the updates to your distribution have been deployed, the links you create in the following steps might not work.
4. Using the method provided by your DNS service provider, add a CNAME resource record set to the hosted zone for your domain. This new CNAME resource record set will redirect DNS queries from your domain (for example, www.example.com) to the CloudFront domain name for your distribution

(for example, d111111abcdef8.cloudfront.net). For more information, see the documentation provided by your DNS service provider.

> **Note**
> If you're using Route 53 as your DNS service, go to Creating, Changing, and Deleting Resource Record Sets in the *Amazon Route 53 Developer Guide* for information about how to add a CNAME resource record set.

> **Important**
> If you already have an existing CNAME record for your domain name, update that resource record set or replace it with a new one that points to the CloudFront domain name for your distribution.
> In addition, confirm that your CNAME resource record set points to your distribution's domain name and not to one of your origin servers.

5. Using dig or a similar tool, confirm that the CNAME resource record set that you created in Step 4 points to the domain name for your distribution. For more information about dig, go to http://www.kloth.net/services/dig.php.

   The following example shows a dig request on the images.example.com domain, as well as the relevant part of the response.

   ```
   [prompt]> dig images.example.com

   ; <<>> DiG 9.3.3rc2 <<>> images.example.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.example.com.      IN    A

   ;; ANSWER SECTION:
   images.example.com. 10800 IN  CNAME  d111111abcdef8.cloudfront.net.
   ...
   ...
   ```

   The line in the Answer Section shows a CNAME resource record set that routes queries for images.example.com to the CloudFront distribution domain name d111111abcdef8.cloudfront.net. The CNAME resource record set is configured correctly if the name on the right side of `CNAME` is the domain name for your CloudFront distribution. If that is any other value, for example, the domain name for your Amazon S3 bucket, then the CNAME resource record set is configured incorrectly. In that case, go back to Step 4 and correct the CNAME record to point to the domain name for your distribution.

6. Test the alternate domain name by creating some test links that use your domain name in the URL instead of the CloudFront domain name for your distribution.

7. In your application, change the links for your objects to use your alternate domain name instead of the domain name of your CloudFront distribution.

Choosing the Price Class for a CloudFront Distribution

CloudFront has edge locations all over the world. Our cost for each edge location varies and, as a result, the price that we charge you varies depending on the edge location from which CloudFront serves your requests.

CloudFront edge locations are grouped into geographic regions, and we've grouped regions into price classes. The default price class includes all regions. Another price class includes most regions (the United States, Europe, Hong Kong and Singapore, and Japan regions) but excludes the most-expensive regions. A third price class includes only the least-expensive regions (the United States and Europe regions).

By default, CloudFront responds to requests for your objects based only on performance: objects are served from the edge location for which latency is lowest for that viewer. If you're willing to accept higher latency for your viewers in some geographic regions in return for lower cost, you can choose a price class that doesn't include all CloudFront regions. Although CloudFront will serve your objects only from the edge locations in that price class, it still serves content from the edge location that has the lowest latency among the edge locations in your selected price class. However, some of your viewers, especially those in geographic regions that are not in your price class, may see higher latency than if your content were being served from all CloudFront edge locations. For example, if you choose the price class that includes only the United States and Europe, viewers in Australia and in Asia may experience higher latency than if you choose the price class that includes Australia and Asia.

If you choose a price class that does not include all edge locations, CloudFront may still occasionally serve requests for your content from an edge location in a region that is not included in your price class. When this happens, you are not charged the rate for the more expensive region from which your objects were served. Instead, you're charged the rate for the least-expensive region in your selected price class.

You can choose a price class when you create a CloudFront distribution, or you can update an existing distribution using the CloudFront console or the CloudFront API. To find the applicable topic about creating or updating a download or a streaming distribution using the CloudFront console or API, see Actions on Distributions (p. 29).

For more information about CloudFront pricing and price classes, go to Amazon CloudFront Pricing.

Listing, Viewing, and Updating CloudFront Distributions

You can use the CloudFront console to list the CloudFront distributions that are associated with your AWS account, view the settings for a distribution, and update most settings.

When you save changes to your distribution configuration, CloudFront starts to propagate the changes to all edge locations. Until your configuration is updated in an edge location, CloudFront continues to serve your content from that location based on the previous configuration. After your configuration is updated in an edge location, CloudFront immediately starts to serve your content from that location based on the new configuration.

Your changes don't propagate to every edge location instantaneously; propagation to all edge locations can take 15 minutes. When propagation is complete, the status of your distribution changes from **InProgress** to **Deployed**. While CloudFront is propagating your changes to edge locations, we cannot determine whether a given edge location is serving your content based on the previous configuration or the new configuration.

To List, View, and Update CloudFront Distributions Using the CloudFront Console

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

2. In the top pane of the CloudFront console, select the distribution that you want to view or update.

    Note
    The top pane lists all of the distributions that are associated with the AWS account that you
    used when you signed in to the CloudFront console.

3. To view or edit streaming distribution settings, skip to Step 4.

    To view or edit settings for a download distribution, perform the following steps.

    a. In the **Distribution Settings** pane, click the tab for the settings that you want to change: **General**,
        Origins, or **Behaviors**.

    b. For general settings, click **Edit**.

        For origins or cache behaviors, click the origin or cache behavior, and click **Edit**.

    c. Enter or update the applicable values. For information about the fields, see the following topics:

        • **General settings:** Distribution Details (p. 39)
        • **Origin settings:** Origin Settings (p. 33)
        • **Cache behavior settings:** Cache Behavior Settings (p. 35)

    d. Click **Yes, Edit**.

4. To edit or view settings for a streaming distribution:

    a. In the **Distribution Details** pane, click **Edit**.

    b. Enter or update the applicable values. For information about the fields, see Values that You
        Specify When You Create or Update a Streaming Distribution (p. 46).

    c. Click **Yes, Edit**.

Deleting a Distribution

If you no longer want to use a distribution, use the following procedure to delete it using the CloudFront
console.

You can also delete a distribution using the CloudFront API:

• To delete a download distribution, use the `DELETE Distribution` API action. For more information,
  go to DELETE Distribution in the *Amazon CloudFront API Reference*.

• To delete a streaming distribution, use the `DELETE Streaming Distribution` API action. For more
  information, go to DELETE Streaming Distribution in the *Amazon CloudFront API Reference*.

    Note
    CloudFront lets you create a combined total of up to 100 download and streaming distributions
    for an AWS account.

To Delete a CloudFront Distribution Using the CloudFront Console

1. Sign in to the AWS Management Console and open the Amazon CloudFront console at
    https://console.aws.amazon.com/cloudfront/.
2. In the right pane of the CloudFront console, find the distribution that you want to delete.
3. If the value of the **State** column is **Disabled**, skip to Step 7.

    If the value of **State** is **Enabled** and the value of **Status** is **Deployed**, continue with Step 4 to disable
    the distribution before deleting it.

    If the value of **State** is **Enabled** and the value of **Status** is **InProgress**, wait until **Status** changes
    to **Deployed**. Then continue with Step 4 to disable the distribution before deleting it.
4. In the right pane of the CloudFront console, check the check box for the distribution that you want
    to delete.
5. Click **Disabled** to disable the distribution, and click **Yes, Disable** to confirm. Then click **Close**.
6. The value of the **State** column immediately changes to **Disabled**. Wait until the value of the **Status**
    column changes to **Deployed**.
7. Check the check box for the distribution that you want to delete.
8. Click **Delete**, and click **Yes, Delete** to confirm. Then click **Close**.

Working with Objects

Topics

This section describes how you work with objects in CloudFront.

Format of URLs for CloudFront Objects

Topics

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 (for example, `www.example.com`), you can add a CNAME alias. For more information, see Using Alternate Domain Names (CNAMEs) (p. 52).

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 on which there are no restrictions.

- Objects in an Amazon S3 bucket that end users must access through CloudFront but that don't require a signed URL. These objects can't be accessed using an Amazon S3 URL.

Signed URLs are required to access the objects that are specified by a cache behavior that you have configured to require signed URLs. Note that if a request for an object (for example, `image.jpg`) matches the path patterns for two or more cache behaviors, CloudFront will process the request based on the cache behavior that is listed first in the distribution. If the first cache behavior doesn't require signed URLs and the second cache behavior does require signed URLs, end users will be able to access `image.jpg` without a signed URL.

For more information about cache behaviors, including path patterns, see Cache Behavior Settings (p. 35). For more information about signed URLs, see Serving Private Content through CloudFront (p. 87).

Format of 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. 51).

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 download distributions, if you're storing your content in more than one Amazon S3 bucket, the format of URLs is the same—URLs don't include any information about your Amazon S3 buckets. To route requests to the applicable bucket, you create an origin for each bucket and create one or more cache behaviors that route requests to each origin. The path pattern in a cache behavior specifies which requests are routed to the origin (the Amazon S3 bucket) that is associated with that cache behavior. For more information about the settings for origins and for cache behaviors in a CloudFront distribution, see Values that You Specify When You Create or Update a Download Distribution (p. 32).

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.

Format of 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
```

If you're storing your content on more than one custom origin, the format of URLs is the same—URLs don't include any information about the custom origin. To route requests to the applicable custom origin, you add an origin to your distribution for each custom origin and create one or more cache behaviors that route requests to each origin. The path pattern in a cache behavior specifies which requests are routed to the origin that is associated with that cache behavior. For more information about the settings for origins and for cache behaviors in a CloudFront distribution, see Values that You Specify When You Create or Update a Download Distribution (p. 32).

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.

Format of 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 Serving Private Content through CloudFront (p. 87).

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. 82).

For information about how to restrict your download distribution so that end users can only access objects using HTTPS, see Using an HTTPS Connection to Access Your Objects (p. 136). (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 Forwards, Caches, and Logs Query String Parameters

For download distributions, you can choose whether you want CloudFront to forward query string parameters to your origin. For streaming distributions, you cannot configure CloudFront to forward query string parameters to your origin.

For both types of distributions, if you enable logging, CloudFront logs the full URL, including query string parameters. For download distributions, this is true regardless of whether you have configured CloudFront to forward query strings. For more information about CloudFront logging, see Access Logs (p. 143).

For more information, see the applicable topic:

- Query String Parameters and Download Distributions (p. 61)
- Query String Parameters and Streaming Distributions (p. 62)

Query String Parameters and Download Distributions

For download distributions, you can specify whether you want CloudFront to include query strings when it forwards requests to your origin. For example, you can specify whether you want CloudFront to forward the *?parameter1=a* part of the following URL:

http://d111111abcdef8.cloudfront.net/images/image.jpg*?parameter1=a*

If you configure CloudFront to forward query strings to your origin, CloudFront will include the query string portion of the URL when caching the object. For example, the following query strings cause CloudFront to cache three objects. This is true even if your origin always returns the same `image.jpg` regardless of the query string:

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

If your origin returns different versions of an object (for example, `/images/image.jpg`) based on the query string, select **Yes** for **Forward Query Strings** in the CloudFront console or specify `true` for the value of the `QueryString` element in the `DistributionConfig` complex type when you're using the CloudFront API.

If your origin returns the same version of an object regardless of the query string, select **No** or `false`. This increases the likelihood that CloudFront can serve a request from the cache, which improves performance and reduces the load on your origin.

The order of parameters matters in query strings. If you configure CloudFront to forward query strings to your origin, the following query strings cause CloudFront to cache two objects:

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

Case also matters in query strings. If you configure CloudFront to forward query strings to your origin, the following query strings cause CloudFront to cache two objects:

- `http://d111111abcdef8.cloudfront.net/images/image.jpg`*?parameter1=a*
- `http://d111111abcdef8.cloudfront.net/images/image.jpg`*?parameter1=A*

If you're using signed URLs to restrict access to your content (if you added trusted signers to your distribution), CloudFront removes the following query string parameters before forwarding the rest of the URL to your origin:

- `Expires`
- `Key-Pair-Id`
- `Policy`
- `Signature`

This means that if you're using signed URLs and you're configuring CloudFront to forward query string parameters to your origin, your own query string parameters cannot be named `Expires`, `Key-Pair-Id`, `Policy`, or `Signature`.

Query String Parameters and Streaming Distributions

For streaming distributions, when CloudFront requests an object from the origin server, it removes any query string parameters. For example, if CloudFront receives the following request and `media.flv` is not already in the CloudFront cache:

`http://d111111abcdef8.cloudfront.net/media/media.flv`*?parameter1=a*

it sends the following URL to your origin server:

`http://d111111abcdef8.cloudfront.net/media/media.flv`

How CloudFront Forwards, Caches, and Logs Cookies

For HTTP and HTTPS download distributions, you can choose whether you want CloudFront to forward cookies to your origin. For RTMP streaming distributions, you cannot configure CloudFront to process cookies.

Important
Amazon S3 and some HTTP servers do not process cookies. Do not configure CloudFront cache behaviors to forward cookies to an origin that doesn't process cookies, or you'll adversely affect cacheability and, therefore, performance.

Cookies and Download Distributions

For download distributions, you can choose whether you want CloudFront to forward cookies to your origin and, if so, you can choose whether you want CloudFront to forward all cookies or only the cookies that you specify. If you choose to forward only the cookies that you specify, you can specify up to 10 cookies for each cache behavior in your distribution. To request an increase in the default limit of 10 cookies per cache behavior, go to http://aws.amazon.com/cloudfront-request/.

When you configure CloudFront to forward cookies to your origin:

- CloudFront includes cookie name-value pairs, but not cookie attributes (for example, path or expiration time), when it forwards requests to your origin. For example, in the following `Cookie` header, CloudFront forwards the cookie name-value pair `country=ata` but does not forward the cookie attribute `$Path=/`:

  ```
  Cookie: country=ata; $Path=/;
  ```
- If you configure CloudFront to forward only a specified list of cookies, CloudFront removes any cookies not on that list before forwarding the request to your origin.
- `If-Modified-Since` and `If-None-Match` conditional requests are not supported.
- CloudFront sorts the cookies in natural order by cookie name before forwarding the request to your origin.

When your origin returns cookies to CloudFront:

- If you configure CloudFront to forward all cookies to your origin for a given cache behavior, CloudFront caches the response and all cookies and cookie attributes that are returned by your origin. CloudFront also returns the object and all cookies and cookie attributes to the viewer.
- If you configure CloudFront to forward only specified cookies to your origin for a given cache behavior, CloudFront caches the response, the specified cookies, and the associated cookie attributes. CloudFront also returns the object, including only the specified cookies and cookie attributes, to the viewer.
- If you do not want CloudFront to cache cookies and cookie attributes, configure your origin server to add the following header in responses to CloudFront:

  ```
  no-cache="Set-Cookie"
  ```

If you configure CloudFront to forward cookies to your origin, CloudFront uses the cookies that the viewer includes in the request to uniquely identify an object in the cache. For example, if three requests for the object `/images/image1.jpg` contain three different name-value pairs, CloudFront will cache the same object three times, once for each name-value pair. This is true even if your origin ignores the cookie values in the request and always returns the same `image1.jpg` object to CloudFront. As a result, CloudFront forwards more requests to your origin server for the same object, which slows performance and increases the load on your origin server. If your origin server does not vary its response based on the value of a given cookie, we recommend that you not configure CloudFront to forward that cookie to your origin.

Cookie names and values are both case sensitive. For example, if two cookies for the same object are identical except for case, CloudFront will cache the object twice.

If you configure CloudFront to log requests and to log cookies, CloudFront logs all cookies and all cookie attributes even if you configure CloudFront not to forward cookies to your origin or you configure CloudFront to forward only a specified list of cookies. For more information about CloudFront logging, see Access Logs (p. 143).

For information about using the CloudFront console to update a distribution so CloudFront forwards cookies to the origin, see Listing, Viewing, and Updating CloudFront Distributions (p. 55). For information

about using the CloudFront API to update a distribution, go to PUT Distribution Config in the *Amazon CloudFront API Reference.*

Cookies and Streaming Distributions

For streaming distributions, when CloudFront requests an object from the origin server, it removes any cookies before forwarding the request to your origin. If your origin returns any cookies along with the object, CloudFront removes them before returning the object to the viewer. For streaming distributions, CloudFront does not cache cookies in edge caches.

You cannot configure CloudFront to log cookies for streaming distributions.

Adding, Removing, or Replacing Objects in a Distribution

For information about adding objects to a distribution, see Adding Objects to a CloudFront Distribution (p. 64).

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. 64). You can also replace objects with objects that have the same name. See Updating Existing Objects Using the Same Object Names (p. 65). 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 How Long Objects Stay in a CloudFront Edge Cache (Object Expiration) (p. 65).

If you need to quickly remove objects from a distribution, you can invalidate them. For more information, see Invalidating Objects (Download Distributions Only) (p. 67).

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 How Long Objects Stay in a CloudFront Edge Cache (Object Expiration) (p. 65).

> **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 How Long Objects Stay in a CloudFront Edge Cache (Object Expiration) (p. 65).
- 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 How Long Objects Stay in a CloudFront Edge Cache (Object Expiration) (p. 65).) 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 How Long Objects Stay in a CloudFront Edge Cache (Object Expiration)

You can control how long your objects stay in a CloudFront cache before CloudFront forwards another request to your origin. Reducing the duration allows you to serve dynamic content. Increasing the duration means your customers get better performance because your objects are more likely to be served directly from the edge cache. A longer duration also reduces the load on your origin.

Typically, CloudFront serves an object from an edge location until the object 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 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 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, configure your origin to add a value for either the `Cache-Control max-age` directive or the `Expires` header field to each object:

- 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 and the maximum is in the year 2038.

- 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`

  > **Important**
  > 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.

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

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 following sections in *RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1*:

- Section 14.9 Cache Control
- Section 14.21 Expires

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.

Specifying the Minimum Time that CloudFront Caches Objects for Download Distributions

For download distributions, if you are adding `Cache-Control` or `Expires` headers to your objects, you can also specify the minimum amount of time that CloudFront keeps an object in the cache before forwarding another request to the origin. Here's how object headers and a minimum TTL control how long objects remain in the CloudFront cache:

	Minimum TTL = 0 (Default)	Minimum TTL > 0
Origin adds `Cache-Control max-age` directive to objects	Objects are cached for the value of the `Cache-Control max-age` directive	Objects are cached for the greater of the value of the `Cache-Control max-age` directive or the value of the CloudFront Minimum TTL
Origin does not add `Cache-Control max-age` directive	Objects are cached for 24 hours.	Objects are cached for 24 hours.

	Minimum TTL = 0 (Default)	Minimum TTL > 0
Origin adds `Expires` header	Objects are cached until the date in the `Expires` header. After the date passes, CloudFront forwards every request to the origin.	Objects are cached until the date in the `Expires` header. After the date passes, objects are cached for the value of the CloudFront Minimum TTL.
Origin adds `Cache-Control` `no-cache`, `no-store`, and/or `private` directives to objects	CloudFront respects the headers	Objects are cached for the Minimum TTL

For information about how to change settings for download distributions using the CloudFront console, see Listing, Viewing, and Updating CloudFront Distributions (p. 55). For information about how to change settings for download distributions using the CloudFront API, see PUT Config.

Adding Headers to Your Objects Using the Amazon S3 Console

To add a `Cache-Control` or `Expires` header field to Amazon S3 objects 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. In the **Objects and Folders** pane, select the first object to which you want to add a header field.
4. At the top of the **Objects and Folders** pane, click **Actions** and click **Properties**.
5. In the **Properties** pane, click the **Metadata** tab.
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:

     `max-age=`*number of seconds that you want objects to stay in a CloudFront edge cache*
   - For an **Expires** field, enter a date and time in HTML format.

9. Click **Save**.
10. If you want to add a header field to additional objects, in the **Objects and Folders** pane, click the name of the next object, and repeat Steps 6 through 9.

Invalidating Objects (Download Distributions Only)

Topics

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. 64).

> **Important**
> You can only invalidate objects that are served by a download distribution. You cannot invalidate objects that are served by a streaming distribution.

You can invalidate a specified number of objects each month for free. Above that limit, you pay a fee for each object that you invalidate. 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. 71).

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 back 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. 64).

Determining Which Objects to Invalidate

If you want to invalidate all of your objects in all CloudFront edge caches but your users don't necessarily access every object on your origin, you can determine which objects viewers have requested from CloudFront and invalidate only those objects. To determine which objects viewers have requested, enable CloudFront access logging. For more information about access logs, see Access Logs (p. 143).

Invalidating Objects and Displaying Information about Invalidations

You can use the CloudFront console or CloudFront API actions to create and run an invalidation, display a list of the invalidations that you submitted previously, and display detailed information about an individual invalidation. You can also copy an existing invalidation, edit the list of object paths, and run the edited invalidation.

See the applicable topic:

Invalidating Objects Using the CloudFront Console

You can create any number of invalidations, but you can have only three invalidation per distribution in progress at one time. Each request can contain up to 1000 objects to invalidate. If you exceed these limits, the CloudFront console displays an error message. To determine how many invalidations are currently in progress, see the **Status** column on the **Invalidations** tab.

To invalidate objects using the CloudFront console

1. Sign in to the AWS Management Console and open the Amazon CloudFront console at https://console.aws.amazon.com/cloudfront/.
2. Click the distribution for which you want to invalidate objects.
3. Click **Distribution Settings**.
4. Click the **Invalidations** tab.
5. Click **Create Invalidation**.
6. Enter the paths of the objects that you want to invalidate. Note the following:

   - You can only invalidate objects that are associated with a download distribution.

   - You must explicitly invalidate every object and every directory that you want CloudFront to stop serving. You cannot use wildcards to invalidate groups of objects, and you cannot invalidate all of the objects in a directory by specifying the directory path.

   - The path is relative to the distribution. A leading / is optional. For example, to invalidate the object at `http://d111111abcdef8.cloudfront.net/images/image2.jpg`, you would specify:

     `/images/image2.jpg`

     or

     `images/image2.jpg`

   - If you configured CloudFront to forward query strings to your origin, you must include the query strings when invalidating objects, for example:
     - `images/image.jpg?parameter1=a`
     - `images/image.jpg?parameter1=b`

     If client requests include five different query strings for the same object, you must invalidate that object five times, once for each query string. For more information, see How CloudFront Forwards, Caches, and Logs Query String Parameters (p. 61). To determine which query strings are in use, you can enable CloudFront logging. For more information, see Access Logs (p. 143).

   - If the object is a directory and if you have not standardized on a method for specifying directories—with or without a trailing slash (/)—we recommend that you invalidate the directory both with and without a trailing slash, for example, `images` and `images/`. For more information, see How Public URLs Affect the Invalidation of Directories (p. 60).

   - You can specify up to 1000 objects.

   - To invalidate the default root object, specify the path the same way you specify the path for any other object.

- If the path includes non-ASCII characters or unsafe characters as defined in RFC 1783 (http://www.ietf.org/rfc/rfc1738.txt), URL-encode those characters. Do not URL-encode any other characters in the path, or CloudFront will not invalidate the old version of the updated object.

7. Click **Invalidate**.

Copying, Editing, and Rerunning an Existing Invalidation Using the CloudFront Console

You can copy an invalidation that you created previously, update the list of object paths, and run the updated validation. You cannot copy an existing invalidation, update the object paths, and save the updated invalidation without running it.

> **Important**
> If you copy an invalidation that is still in progress, update the list of object paths, and run the updated invalidation, CloudFront will not stop or delete the invalidation that you copied. If any object paths appear in the original and in the copy, CloudFront will try to invalidate the objects twice, and both invalidations will count against your maximum number of free invalidations for the month. If you've already reached the maximum number of free validations, you'll be charged for both invalidations of each object. For more information, see Invalidation Limits (p. 71)

To copy, edit, and rerun an existing invalidation using the CloudFront console

1. Sign in to the AWS Management Console and open the Amazon CloudFront console at https://console.aws.amazon.com/cloudfront/.
2. Click the distribution that contains the invalidation that you want to copy.
3. Click **Distribution Settings**.
4. Click the **Invalidations** tab.
5. Click the invalidation that you want to copy.

   If you aren't sure which invalidation you want to copy, you can click an invalidation and click **Details** to display detailed information about that invalidation.
6. Click **Copy**.
7. Update the list of object paths if applicable.
8. Click **Invalidate**.

Listing Invalidations Using the CloudFront Console

Using the console, you can display a list of the last 100 invalidations that you've created and run for a distribution. If you want to get a list of more than 100 invalidations, use the GET Invalidation List API action. For more information, go to GET Invalidation List in the *Amazon CloudFront API Reference*.

To list invalidations using the CloudFront console

1. Sign in to the AWS Management Console and open the Amazon CloudFront console at https://console.aws.amazon.com/cloudfront/.
2. Click the distribution for which you want to display a list of invalidations.
3. Click **Distribution Settings**.
4. Click the **Invalidations** tab.

Displaying Information About an Invalidation Using the CloudFront Console

You can display detailed information about an invalidation, including distribution ID, invalidation ID, the status of the invalidation, the date and time that the invalidation was created, and a complete list of the object paths.

To display information about an invalidation using the CloudFront console

1. Sign in to the AWS Management Console and open the Amazon CloudFront console at https://console.aws.amazon.com/cloudfront/.
2. Click the distribution that contains the invalidation about which you want to display detailed information.
3. Click **Distribution Settings**.
4. Click the **Invalidations** tab.
5. Click the invalidation about which you want to display detailed information.
6. Click **Details**.

Invalidating Objects and Displaying Information about Invalidations Using the CloudFront API

For information about invalidating objects and about displaying information about invalidations using the CloudFront API, see the applicable topic in the *Amazon CloudFront API Reference*:

- Invalidating objects: POST Invalidation
- Getting a list of your invalidations: GET Invalidation List
- Getting information about a specific invalidation: GET Invalidation

Third-Party Tools for Invalidating Objects

In addition to the invalidation methods provided by CloudFront, several third-party tools provide ways to invalidate objects. For a list of tools, see Invalidating Objects (p. 253).

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 1000 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 1000 object invalidations you request per month are free; you pay for each object invalidation over 1000 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 and Displaying Information about Invalidations (p. 68).

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

For a large object, 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.

When CloudFront receives a `Range GET` request, it checks the cache in the edge location that received the request. If the cache in that edge location already contains the entire object or the requested portion of the object, CloudFront immediately serves the requested range from the cache. If the cache doesn't contain the requested range, CloudFront forwards the request to the origin. To optimize performance, CloudFront may request a larger range than the client requested in the `Range GET`.

If the origin supports `Range GET` requests, it returns the requested range. CloudFront serves the requested range and also caches it for future requests. (Amazon S3 supports `Range GET` requests, as do some HTTP servers, for example, Apache and IIS. For information about whether your HTTP server does, go to the documentation for your HTTP server.)

If the origin doesn't support `Range GET` requests, it returns the entire object. CloudFront serves the entire object and also caches the entire object for future requests. After CloudFront caches the entire object in an edge cache, it responds to `Range GET` requests by serving the requested range.

CloudFront generally follows the RFC specification for the `Range` header. However, if your `Range` headers don't adhere to the following requirements, CloudFront will return HTTP status code 200 with the full object instead of status code 206 with the specified ranges:

- The ranges must be listed in ascending order. For example, `100-200,300-400` is valid, `300-400,100-200` is not valid.
- The ranges must not overlap. For example, `100-200,150-250` is not valid.
- All of the ranges specifications must be valid. For example, you can't specify a negative value as part of a range.

For more information about the `Range` request header, see "Section 14.35 Range" in *Hypertext Transfer Protocol -- HTTP/1.1* at http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35.

Specifying a Default Root Object (Download Distributions Only)

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 Serving Private Content through CloudFront (p. 87).
- **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 download distribution or returning an error, perform the following procedure to 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, go to the *Amazon Simple Storage Service Console User Guide*.

3. Update your distribution to refer to the default root object using the CloudFront console or the CloudFront API.

   To specify a default root object using the CloudFront console:

   a. Sign in to the AWS Management Console and open the Amazon CloudFront console at https://console.aws.amazon.com/cloudfront/.
   b. In the list of distributions in the top pane, select the distribution to update.
   c. In the **Distribution Details** pane, on the **General** tab, click **Edit**.
   d. In the **Edit Distribution** dialog box, in the **Default Root Object** field, enter the file name of the default root object.
   e. To save your changes, click **Yes, Edit**.

   To update your configuration using the CloudFront API, you specify a value for the `DefaultRootObject` element in your distribution. For information about using the CloudFront API to specify a default root object, see PUT Distribution Config 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 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 distribution 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. 77).

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` in the request header. The content must be compressed using gzip; other compression algorithms are not supported. If the request header includes additional content encodings, for example, `deflate` or `sdch`, CloudFront removes them before forwarding the request to the origin server. If `gzip` 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.

For more information, see the following topics:

Topics

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. 77).
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` 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.

In addition, if you have compressed objects that are requested less frequently than every few seconds, you may have to change the values of `frequentHitThreshold` and `frequentHitTimePeriod`.

For more information, refer to the IIS documentation on the Microsoft website.

Serving Compressed Files When Your Origin Server Is Running NGINX

Some versions of NGINX require that you customize NGINX settings when you're using CloudFront to serve compressed files. In the documentation for your version of NGINX, see the documentation for the `HttpGzipModule` for more information about the following settings:

- **gzip_http_version:** CloudFront sends requests in HTTP 1.0 format. In some versions of NGINX, the default value for the `gzip_http_version` setting is 1.1. If your version of NGINX includes this setting, change the value to `1.0`.

- **gzip_proxied:** When CloudFront forwards a request to the origin server, it includes a `Via` header. This causes NGINX to interpret the request as proxied and, by default, NGINX disables compression for proxied requests. If your version of NGINX includes the `gzip_proxied` setting, change the value to `any`.

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`.

   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 Objects and Folders 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`.
   h. Click **Save**.
   i. Repeat Step 4d through 4h 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` 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. 205). For more information about CloudFront private content, see Serving Private Content through CloudFront (p. 87).

Request and Response Behavior, and Supported HTTP Status Codes

See the applicable section:

Topics

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:

Topics

HTTP Methods

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

Query Strings

For download distributions, you can configure whether CloudFront forwards query string parameters to your Amazon S3 origin. For streaming distributions, CloudFront does not forward query string parameters. For more information, see How CloudFront Forwards, Caches, and Logs Query String Parameters (p. 61).

Cookies

Amazon S3 doesn't process cookies. If you configure a cache behavior to forward cookies to an Amazon S3 origin, CloudFront forwards the cookies, but Amazon S3 ignores them.

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

For download distributions, to control how long your objects stay in a CloudFront cache before CloudFront forwards another request to your origin, you can:

- Configure your origin to add a `Cache-Control` or an `Expires` header field to each object.
- Specify a value for Minimum TTL in CloudFront cache behaviors.
- Use the default value of 24 hours.

For more information, see Specifying How Long Objects Stay in a CloudFront Edge Cache (Object Expiration) (p. 65).

Conditional GETs

When CloudFront receives a request for an object that has expired from an edge cache, it forwards the request to the Amazon S3 origin either to get the latest version of the object or to get confirmation from Amazon S3 that the CloudFront edge cache already has the latest version. When Amazon S3 originally sent the object to CloudFront, it included an `ETag` value and a `LastModified` value in the response. In the new request that CloudFront forwards to Amazon S3, CloudFront adds one of the following:

- An `If-Match` or `If-None-Match` header that contains the `ETag` value for the expired version of the object.
- An `If-Modified-Since` header that contains the `LastModified` value for the expired version of the object.

Amazon S3 uses this information to determine whether the object has been updated and, therefore, whether to return the entire object to CloudFront or to return only an HTTP 304 status code (not modified).

Maximum Length of a Request and Maximum Length of a URL

The maximum length of a request, including the path, the query string (if any), and headers, is 20480 bytes.

CloudFront constructs a URL from the request. The maximum length of this URL is 8192 bytes.

If a request or a URL exceeds these limits, CloudFront drops the request.

How CloudFront Processes Responses from Your Amazon S3 Origin Server

Topics

Redirects

You can configure an Amazon S3 bucket to redirect all requests to another host name; this can be another Amazon S3 bucket or an HTTP server. If you configure a bucket to redirect all requests and if the bucket is the origin for a CloudFront distribution, we recommend that you configure the bucket to redirect all requests to a CloudFront distribution using either the domain name for the distribution (for example, d111111abcdef8.cloudfront.net) or an alternate domain name (a CNAME) that is associated with a distribution (for example, example.com). Otherwise, end user requests bypass CloudFront, and the objects are served directly from the new origin.

> **Note**
> If you redirect requests to an alternate domain name, you must also update the DNS service for your domain by adding a CNAME record. For more information, see Using Alternate Domain Names (CNAMEs) (p. 52).

Here's what happens when you configure a bucket to redirect all requests:

1. A viewer (for example, a browser) requests an object from CloudFront.
2. CloudFront forwards the request to the Amazon S3 bucket that is the origin for your distribution.
3. Amazon S3 returns an HTTP status code 301 (Moved Permanently) as well as the new location.
4. CloudFront caches the redirect status code and the new location, and returns the values to the viewer. CloudFront does not follow the redirect to get the object from the new location.
5. The viewer sends another request for the object, but this time the viewer specifies the new location that it got from CloudFront:

    - If the Amazon S3 bucket is redirecting all requests to a CloudFront distribution, using either the domain name for the distribution or an alternate domain name, CloudFront requests the object from the Amazon S3 bucket or the HTTP server in the new location. When the new location returns the object, CloudFront returns it to the viewer and caches it in an edge location.

    - If the Amazon S3 bucket is redirecting requests to another location, the second request bypasses CloudFront. The Amazon S3 bucket or the HTTP server in the new location returns the object directly to the viewer, so the object is never cached in a CloudFront edge cache.

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.

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:

Topics

HTTP Methods

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

Query Strings

You can configure whether CloudFront forwards query string parameters to your origin. For more information, see How CloudFront Forwards, Caches, and Logs Query String Parameters (p. 61).

Cookies

You can configure CloudFront to forward cookies to your origin. For more information, see How CloudFront Forwards, Caches, and Logs Cookies (p. 62).

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 SSLv3 or TLSv1 protocols and the AES128-SHA1 or RC4-MD5 ciphers.

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 value of the **Origin Protocol Policy** field in the CloudFront console or, if you're using the CloudFront API, the `OriginProtocolPolicy` element in the `DistributionConfig` complex type. In the CloudFront console, the options are **HTTP Only** and **Match Viewer**.

If you specify **HTTP Only**, 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**, CloudFront forwards requests to the origin server using the protocol in the end-user request. Note that CloudFront caches the object only once even if viewers make requests using both HTTP and HTTPS protocols.

> **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.

If you aren't sure which protocol to use, we recommend that you specify HTTP only.

For information about how to update a distribution using the CloudFront console, see Listing, Viewing, and Updating CloudFront Distributions (p. 55). For information about how to update a distribution using the CloudFront API, go to PUT Distribution Config in the *Amazon CloudFront API Reference*.

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"`.

For more information, see Serving Compressed Files (p. 74).

Caching Duration and Minimum TTL

For download distributions, to control how long your objects stay in a CloudFront cache before CloudFront forwards another request to your origin, you can:

- Configure your origin to add a `Cache-Control` or an `Expires` header field to each object.
- Specify a value for Minimum TTL in CloudFront cache behaviors.
- Use the default value of 24 hours.

For more information, see Specifying How Long Objects Stay in a CloudFront Edge Cache (Object Expiration) (p. 65).

Conditional GETs

When CloudFront receives a request for an object that has expired from an edge cache, it forwards the request to the origin either to get the latest version of the object or to get confirmation from the origin that the CloudFront edge cache already has the latest version. Typically, when the origin last sent the object to CloudFront, it included an `ETag` value, a `LastModified` value, or both values in the response. In the new request that CloudFront forwards to the origin, CloudFront adds one of the following:

- An `If-Match` or `If-None-Match` header that contains the `ETag` value for the expired version of the object.
- An `If-Modified-Since` header that contains the `LastModified` value for the expired version of the object.

The origin uses this information to determine whether the object has been updated and, therefore, whether to return the entire object to CloudFront or to return only an HTTP 304 status code (not modified).

Maximum Length of a Request and Maximum Length of a URL

The maximum length of a request, including the path, the query string (if any), and headers, is 20480 bytes.

CloudFront constructs a URL from the request. The maximum length of this URL is 8192 bytes.

If a request or a URL exceeds these limits, CloudFront drops the request.

How CloudFront Processes Responses from Your Custom Origin Server

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

Topics
- Maximum File Size (p. 84)

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.

Cookies

If you enable cookies for a cache behavior, and if the origin returns cookies with an object, CloudFront caches both the object and the cookies. Note that this reduces cacheability for an object. For more information, see How CloudFront Forwards, Caches, and Logs Cookies (p. 62).

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.

Origin Unavailable

If your origin server is unavailable and CloudFront gets a request for an object that is in the edge cache but that has expired (for example, because the period of time specified in the `Cache-Control max-age` directive has passed), CloudFront continues to serve the expired version of the object. For more information about object expiration, see Specifying How Long Objects Stay in a CloudFront Edge Cache (Object Expiration) (p. 65).

In some cases, an object that is seldom requested is evicted and is no longer available in the edge cache. CloudFront can't serve an object that has been evicted.

Transfer Encoding

CloudFront supports only the `chunked` value of the `Transfer-Encoding` header. If your origin returns `Transfer-Encoding: chunked`, CloudFront assembles the chunks into a monolithic object as they're received at an edge location, returns the object to the client, and caches the object for subsequent requests. In addition, CloudFront calculates a `Content-Length` header and returns it in response to subsequent client requests.

We recommend that you not use chunked transfer encoding. If a chunk is delayed (by a firewall, for example), CloudFront has no way to know whether it already has the last chunk in an object, and may close the connection and cache a partial object. For more information, see Dropped TCP Connections (p. 85).

Dropped TCP Connections

If the TCP connection between CloudFront and your origin drops while your origin is returning an object to CloudFront, CloudFront behavior depends on whether your origin included a `Content-Length` header in the response:

- **Content-Length header:** CloudFront returns the object to the viewer as it gets the object from your origin. However, if the value of the `Content-Length` header doesn't match the size of the object, CloudFront doesn't cache the object.
- **No Content-Length header:** CloudFront returns the object to the viewer and caches it, but the object may not be complete. Without a `Content-Length` header, CloudFront cannot determine whether the TCP connection was dropped accidentally or on purpose.

We recommend that you configure your HTTP server to add a `Content-Length` header to prevent CloudFront from caching partial objects.

Supported HTTP Status Codes for Custom Origin Servers

In general, if your custom origin server responds to a CloudFront request with any of the following status codes, CloudFront caches the code for five minutes regardless of the minimum TTL setting for the applicable cache behavior. In addition, CloudFront writes the results to the access logs.

If CloudFront already has an expired version of the requested object in the edge cache, it serves the expired object instead.

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

Serving Private Content through CloudFront

Many companies that distribute content via the Internet want to restrict access to documents, business data, media streams, or content that is intended for selected users, for example, users who have paid a fee. To securely serve this private content using CloudFront, you can:

- Require that your users use special CloudFront signed URLs to access your content, not the standard CloudFront public URLs.
- Require that your users access your Amazon S3 content using CloudFront URLs, not Amazon S3 URLs.

Overview of Private Content

You can control end-user access to your private content in two ways:

❶	**You can restrict access to objects in CloudFront edge caches:** You can configure CloudFront to require that end users access your objects using special **signed URLs**. You then create the signed URLs (either manually or programmatically) and distribute them to your users. When you create signed URLs for your objects, you can specify: • An ending date and time, after which the URL is no longer valid. • (Optional) The date and time that the URL becomes valid. • (Optional) The IP address or range of addresses of the computers that can be used to access your content. One part of a signed URL is hashed and signed using the private key from a public/private key pair. When someone uses a signed URL to access an object, CloudFront compares the signed and unsigned portions of the URL. If they don't match, CloudFront doesn't serve the object.

> **2** **You can restrict access to objects in your Amazon S3 bucket:** You can secure the content in your Amazon S3 bucket so end users can access it using CloudFront URLs but cannot access it using Amazon S3 URLs. This prevents anyone from bypassing CloudFront and using the Amazon S3 URL to access the content to which you're trying to restrict access. To require that users use CloudFront URLs, you:
>
> - Create a special CloudFront user called an **origin access identity**.
> - Give the origin access identity permission to read the objects in your bucket.
> - Remove permission for anyone else to read the objects.
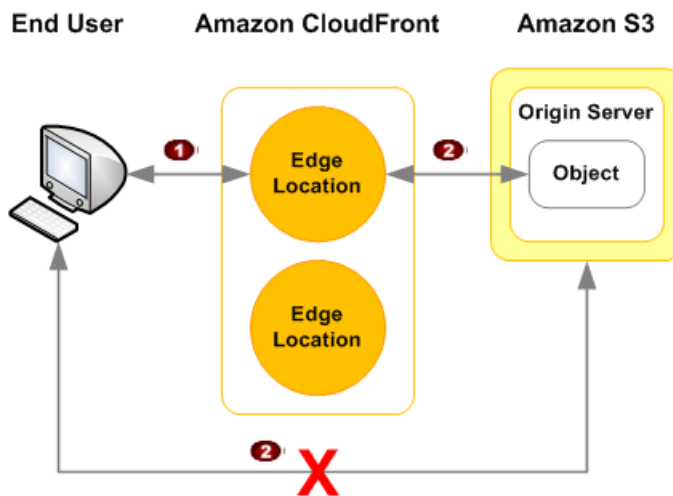
How Private Content Works

Here's an overview of how you use private content to secure access to your Amazon S3 content. Later, we go into greater detail for each step.

> **Note**
> To set up private content, you must use the CloudFront console or CloudFront API version 2009-09-09 or later.

1. You secure your content in Amazon S3 to prevent anyone from bypassing CloudFront and using the Amazon S3 URL to access the content to which you're trying to restrict access. This step is optional, but it's a good idea in case someone learns the Amazon S3 URLs for your content.

   a. Create an origin access identity, which is a special CloudFront user.
   b. Associate the origin access identity with your distribution. (For download distributions, you associate the origin access identity with origins, so you can secure all or just some of your Amazon S3 content.)
   c. Change permissions in Amazon S3 so only the origin access identity can access your objects.

   For more information, see Using an Origin Access Identity to Restrict Access to Your Amazon S3 Content (p. 92).

2. In your CloudFront distribution, specify one or more trusted signers, which are the AWS accounts that you want to have permission to create signed URLs.

For more information, see Specifying the AWS Accounts That Can Create Signed URLs (Trusted Signers) (p. 97).

3. You develop your application to create signed URLs for the objects or parts of your application for which signed URLs are required.

    For more information about signed URLs, see Overview of Signed URLs (p. 102).

4. An end user requests an object for which you want to require signed URLs.

5. Your application verifies that the end user is entitled to access the object: they've signed in, they've paid for access to the content, or they've met some other requirement for access.

6. Your application creates and returns a signed URL to the user.

7. The signed URL allows the user to download or stream the content.

    This step is automatic; the user usually doesn't have to do anything additional to access the content. For example, if a user is accessing your content in a web browser, CloudFront returns the signed URL to the browser. The browser immediately uses the signed URL to access the object in the CloudFront edge cache without any intervention from the user.

8. CloudFront confirms that the URL hasn't been tampered with, and performs the standard CloudFront operations: determines whether the object is already in the edge cache, forwards the request to the origin if necessary, and returns the object to the user.

Using an HTTP Server for Private Content

You can use signed URLs for any CloudFront distribution, regardless of whether the origin is an Amazon S3 bucket or an HTTP server. However, for CloudFront to access your objects on an HTTP server, the objects must remain publicly accessible. Because the objects are publicly accessible, anyone who has the URL for an object on your HTTP server can access the object without the protection provided by CloudFront signed URLs. If you use signed URLs and your origin is an HTTP server, do not give the URLs for the objects on your HTTP server to your customers or to others outside your organization.

Choosing How Long Signed URLs Are Valid

You can distribute private content using a signed URL that is valid for only a short time—possibly for as little as a few minutes. Signed URLs that are valid for such a short period are good for distributing content on-the-fly to an end user for a limited purpose, such as distributing movie rentals or music downloads to customers on demand. If your signed URLs will be valid for just a short period, you'll probably want to generate them automatically using an application that you develop. When the user starts to download an object or starts to play a media file, CloudFront compares the expiration time in the URL with the current time to determine whether the URL is still valid.

You can also distribute private content using a signed URL that is valid for a longer time, possibly for years. Signed URLs that are valid for a longer period are useful for distributing private content to known end users, such as distributing a business plan to investors or distributing training materials to employees. You can develop an application to generate these longer-term signed URLs for you, or you can use one of the third-party GUI tools listed in Tools for Configuring Private Content (p. 254).

Sample Code and Third-Party Tools

For sample code that creates the hashed and signed part of signed URLs, see the following topics:

*   Create a URL Signature Using Perl (p. 121)
*   Create a URL Signature Using PHP (p. 122)
*   Create a URL Signature Using C# and the .NET Framework (p. 125)
*   Create a URL Signature Using Java (p. 133)

Additional sample code for creating signed URLs is available on the Amazon CloudFront Sample Code & Libraries page.

For information about third-party tools that support private content, including creating signed URLs, see Tools for Configuring Private Content (p. 254).

Task List: Serving Private Content

To configure CloudFront to serve private content, perform the following tasks.

1. *Optional:* Configure your CloudFront distribution and your Amazon S3 bucket to require that your users access your Amazon S3 content using only CloudFront URLs. For more information, see Using an Origin Access Identity to Restrict Access to Your Amazon S3 Content (p. 92).
2. Specify the AWS accounts that you want to use to create signed URLs. For more information, see Specifying the AWS Accounts That Can Create Signed URLs (Trusted Signers) (p. 97).
3. Create signed URLs, either manually or programmatically, to give to authorized end users. For more information, see Overview of Signed URLs (p. 102).

Using an Origin Access Identity to Restrict Access to Your Amazon S3 Content

Topics

Typically, if you're using an Amazon S3 bucket as the origin for a CloudFront distribution, you grant everyone permission to read the objects in your bucket. This allows anyone to access your objects using either the CloudFront URL or the Amazon S3 URL. CloudFront doesn't expose Amazon S3 URLs, but your users may have those URLs if your application serves any objects directly from Amazon S3 or if anyone gives out direct links to specific objects in Amazon S3.

If you want to use CloudFront signed URLs to provide access to objects in your Amazon S3 bucket, you probably also want to prevent users from accessing your Amazon S3 objects using Amazon S3 URLs. If users access your objects directly in Amazon S3, they bypass the controls provided by CloudFront signed URLs, including control over when a URL expires and control over which IP addresses can be used to access the objects. In addition, if users access objects using both CloudFront URLs and Amazon S3 URLs, CloudFront access logs are less useful because they're incomplete.

You restrict access to Amazon S3 content by creating an origin access identity, which is a special CloudFront user. You change Amazon S3 permissions to give the origin access identity permission to access your objects, and to remove permissions from everyone else. When your users access your Amazon S3 objects using CloudFront URLs, the CloudFront origin access identity gets the objects on your users' behalf. If your users try to access objects using Amazon S3 URLs, they're denied access. The origin access identity has permission to access objects in your Amazon S3 bucket, but users don't.

> **Note**
> To create origin access identities, you must use the CloudFront console or CloudFront API version 2009-09-09 or later.

To ensure that your users access your objects using only CloudFront URLs, regardless of whether the URLs are signed, perform the following tasks:

1. Create an origin access identity and add it to your distribution. For more information, see Creating a CloudFront Origin Access Identity and Adding it to Your Distribution (p. 92).

   > **Note**
   > You can also create an origin access identity and add it to your distribution when you create the distribution.

2. Change the permissions either on your Amazon S3 bucket or on the objects in your bucket so only the origin access identity has read permission (or read and download permission).

   For more information, see Granting the Origin Access Identity Permission to Read Objects in Your Amazon S3 Bucket (p. 94).

Creating a CloudFront Origin Access Identity and Adding it to Your Distribution

An AWS account can have up to 100 CloudFront origin access identities. However, you can add an origin access identity to as many distributions as you want, so one origin access identity is usually sufficient.

Amazon CloudFront Developer Guide
Creating a CloudFront Origin Access Identity and Adding
it to Your Distribution

If you didn't create an origin access identity and add it to your distribution when you created the distribution, you can create and add one now using either the CloudFront console or the CloudFront API:

- **If you're using the CloudFront console:** You can create an origin access identity and add it to your distribution at the same time. For more information, see Creating an Origin Access Identity and Adding it to Your Distribution Using the CloudFront Console (p. 93).
- **If you're using the CloudFront API:** You create an origin access identity and then you add it to your distribution. Perform the procedure in each of the following topics:
  - Creating an Origin Access Identity Using the CloudFront API (p. 94)
  - Adding an Origin Access Identity to Your Distribution Using the CloudFront API (p. 94)

Creating an Origin Access Identity and Adding it to Your Distribution Using the CloudFront Console

If you didn't create an origin access identity when you created your distribution, perform the following procedure.

To create a CloudFront origin access identity using the CloudFront console

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

2. Click the icon for the distribution to which you want to add an origin access identity.

3. Change to edit mode:

   - **Download distributions:** Click the **Origins** tab, click the origin that you want to edit, and click **Edit**. You can only create an origin access identity for origins for which **Origin Type** is **S3 Origin**.
   - **Streaming distributions:** Click **Edit**.

4. For **Restrict Bucket Access**, click **Yes**.

5. If you already have an origin access identity that you want to use, click **Use an Existing Identity**. Then select the identity in the **Your Identities** list.

   > **Note**
   > If you already have an origin access identity, we recommend that you reuse it to simplify maintenance.

   If you want to create an identity, click **Create a New Identity**. Then enter a description for the identity in the **Comment** field.

6. If you want CloudFront to automatically give the origin access identity permission to read the objects in the Amazon S3 bucket specified in **Origin Domain Name**, click **Yes, Update Bucket Policy**.

   > **Important**
   > If you click **Yes, Update Bucket Policy**, CloudFront updates bucket permissions to grant the specified origin access identity the permission to read objects in your bucket. However, CloudFront does not remove existing permissions. If users currently have permission to access the objects in your bucket using Amazon S3 URLs, they will still have that permission after CloudFront updates your bucket permissions. To view or remove existing bucket permissions, use a method provided by Amazon S3. For more information, see Granting the Origin Access Identity Permission to Read Objects in Your Amazon S3 Bucket (p. 94).

   If you want to manually update permissions on your Amazon S3 bucket, click **No, I Will Update Permissions**.

7. Click **Yes, Edit**.

8.   If you're adding an origin access identity to a download distribution and you have more than one origin, repeat Step 3 through Step 7 as applicable.

Creating an Origin Access Identity Using the CloudFront API

If you already have an origin access identity and you want to reuse it instead of creating another one, skip to Adding an Origin Access Identity to Your Distribution Using the CloudFront API (p. 94).

To create a CloudFront origin access identity using the CloudFront API, use the `POST Origin Access Identity` API action. The response includes an `Id` and an `S3CanonicalUserId` for the new origin access identity. Make note of these values because you will use them later in the process:

- **Id element:** You use the value of the `Id` element to associate an origin access ID with your distribution.
- **S3CanonicalUserId element:** You use the value of the `S3CanonicalUserId` element when you give CloudFront access to your Amazon S3 bucket or objects.

For more information about the `POST Origin Access Identity API` action, go to POST Origin Access Identity in the *Amazon CloudFront API Reference*. For a list of other actions that you can perform on origin access identities, go to Actions on Origin Access Identities, also in the *Amazon CloudFront API Reference*.

Adding an Origin Access Identity to Your Distribution Using the CloudFront API

You can use the CloudFront API to add a CloudFront origin access identity to an existing distribution or to create a new distribution that includes an origin access identity. In either case, include an `OriginAccessIdentity` element. This element contains the value of the `Id` element that the `POST Origin Access Identity` API action returned when you created the origin access identity. For download distributions, add the `OriginAccessIdentity` element to one or more origins. For streaming distributions, add the `OriginAccessIdentity` element to the distribution.

See the applicable topic in the *Amazon CloudFront API Reference*:

- **Create a new download distribution:** POST Distribution
- **Update an existing download distribution:** PUT Distribution Config
- **Create a new streaming distribution:** POST Streaming Distribution
- **Update an existing streaming distribution:** PUT Streaming Distribution Config

Granting the Origin Access Identity Permission to Read Objects in Your Amazon S3 Bucket

When you create or update a distribution, you can add an origin access identity and automatically update the bucket policy to give the origin access identity permission to access your bucket. Alternatively, you can choose to manually change the bucket policy or change ACLs, which control permissions on individual objects in your bucket.

Whichever method you use, you should still review the bucket policy for your bucket and review the permissions on your objects to ensure that:

- CloudFront can access objects in the bucket on behalf of users who are requesting your objects using CloudFront URLs.
- Users can't use Amazon S3 URLs to access your objects.

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.
- By default, your Amazon S3 bucket and all of the objects in it are private—only the AWS account that created the bucket has permission to read or write the objects in it.
- If you're adding an origin access identity to an existing distribution, modify the bucket policy or any object ACLs as appropriate to ensure that the objects are not publicly available.
- Grant additional permissions to one or more secure administrator accounts so you can continue to update the contents of the Amazon S3 bucket.

    Important
    There may be a brief delay between when you save your changes to Amazon S3 permissions and when the changes take effect. Until the changes take effect, you may get permission-denied errors when you try to access objects in your bucket.

Updating Amazon S3 Bucket Policies

You can update the Amazon S3 bucket policy using either the AWS Management Console or the Amazon S3 API:

- Allow the CloudFront origin access identity to access objects in the bucket.

    To specify an origin access identity, use the value of **Amazon S3 Canonical User ID** on the **Origin Access Identity** page in the CloudFront console. If you're using the CloudFront API, use the value of the `S3CanonicalUserId` element that was returned when you created the origin access identity.
- Deny access to anyone that you don't want to have access using Amazon S3 URLs.

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:

- Grant the CloudFront origin access identity `READ` permission on each object that the CloudFront distribution serves.

    To specify an origin access identity, use the value of **Amazon S3 Canonical User ID** on the **Origin Access Identity** page in the CloudFront console. If you're using the CloudFront API, use the value of the `S3CanonicalUserId` element that was returned when you created the origin access identity.
- Deny access to anyone that you don't want to have access using Amazon S3 URLs.

If another AWS account uploads objects to your bucket, that account is the owner of the objects. By default, the account that owns objects in a bucket is the only account that can grant permissions to those objects. However, the AWS account that owns the objects can make you an owner, too, which allows you to change permissions on the objects.

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. 125).

Specifying the AWS Accounts That Can Create Signed URLs (Trusted Signers)

Topics

To create signed URLs for the objects in your Amazon S3 bucket, you need at least one AWS account that has an active CloudFront key pair. This account is known as a trusted signer. The trusted signer has two purposes:

- As soon as you add the AWS account number for your trusted signer to your distribution, CloudFront starts to require that users use signed URLs to access the objects in your Amazon S3 bucket.
- When you create a signed URL, you use the private key from the trusted signer's key pair to sign a portion of the URL. When someone uses the signed URL to access an object, CloudFront compares the signed portion of the URL with the unsigned portion to verify that the URL hasn't been tampered with. CloudFront also verifies that the URL is valid, meaning, for example, that the expiration date and time in the URL hasn't passed.

When you specify trusted signers, you also indirectly specify the objects that require signed URLs:

- **Download distributions:** You add trusted signers to cache behaviors. If your distribution has only one cache behavior, users must use signed URLs to access any object associated with the distribution. If you create multiple cache behaviors and add trusted signers to some cache behaviors and not to others, you can require that users use signed URLs to access some objects and not others.
- **Streaming distributions:** You add trusted signers to a distribution. After you add trusted signers to a streaming distribution, users must use signed URLs to access any object associated with the distribution.

  Note
  To specify trusted signers for a distribution, you must use the CloudFront console or CloudFront API version 2009-09-09 or later.

To specify the accounts that are allowed to create signed URLs and to add the accounts to your CloudFront distribution, perform the following tasks:

1. Decide which AWS accounts you want to use as trusted signers. Most CloudFront customers use the account that they used to created the distribution.
2. For each of the accounts that you selected in Step 1, create a CloudFront key pair. For more information, see Creating CloudFront Key Pairs for Your Trusted Signers (p. 98).
3. If you're using to .NET or Java to create signed URLs, reformat the CloudFront private key. For more information, see Reformatting the CloudFront Private Key (.NET and Java Only) (p. 98).
4. In the distribution for which you're creating signed URLs, specify the AWS account numbers of your trusted signers. For more information, see Adding Trusted Signers to Your Distribution (p. 99).
5. **Optional:** Verify that CloudFront recognizes that your trusted signers have active CloudFront key pairs. When trusted signers have active CloudFront key pairs, CloudFront recognizes them as active trusted signers. For more information, see Verifying that Trusted Signers Are Active (Optional) (p. 101).

Creating CloudFront Key Pairs for Your Trusted Signers

Each of the AWS accounts that you use to create CloudFront signed URLs—your trusted signers—must create its own CloudFront key pair, and the key pair must be active. Note the following:

- IAM users are currently not allowed to create CloudFront key pairs, so you cannot use IAM users as trusted signers.
- You cannot substitute an Amazon EC2 key pair for a CloudFront key pair. When you create a CloudFront signed URL, you include the key pair ID for the trusted signer's key pair in the URL. Amazon EC2 does not make key pair IDs available.

To help secure your applications, we recommend that you change CloudFront key pairs every 90 days or more often. For more information, see Rotating CloudFront Key Pairs (p. 101).

To create CloudFront key pairs for your trusted signers

1. Sign in to the AWS Management Console and open the Security Credentials page at https://portal.aws.amazon.com/gp/aws/securityCredentials.
2. Under **Access Credentials**, click the **Key Pairs** tab.
3. Under **Amazon CloudFront Key Pairs**, click **Create a New Key Pair**.
4. In the **Key Pair Created** dialog box, click **Close**.
5. In the **Opening <filename>** dialog box, accept the default value of **Save File**, and click **OK** to download and save the private key for your CloudFront key pair.

   > **Important**
   > Save the private key for your CloudFront key pair in a secure location, and set permissions on the file so only the desired administrator accounts can read it. If someone gets your private key, they can generate valid signed URLs and download your content. You cannot get the private key again, so if you lose or delete it, you must create a new CloudFront key pair.

6. Record the key pair ID for your key pair. You'll use it when you create signed URLs.

Reformatting the CloudFront Private Key (.NET and Java Only)

If you're using .NET or Java to create signed URLs, you cannot use the private key from your key pair in the default .pem format to create the signature:

- **.NET framework:** Convert the private key to the XML format that the .NET framework uses. Several tools are available. For the example in Create a URL Signature Using C# and the .NET Framework (p. 125), we used .NET 2.0 OpenSSL Public and Private Key Parser, http://www.jensign.com/opensslkey/opensslkey.cs.
- **Java:** Convert the private key to DER format. To do this, you can use OpenSSL:

```
$ openssl pkcs8 -topk8 -nocrypt -in origin.pem -inform PEM -out new.der
-outform DER
```

To ensure that the encoder works correctly, add the jar for the Bouncy Castle Java cryptography APIs to your project and then add the Bouncy Castle provider.

Adding Trusted Signers to Your Distribution

Trusted signers are the AWS accounts that are allowed to create signed URLs for a distribution. By default, no account, not even the account that created the distribution, is allowed to create signed URLs for the distribution. To specify the AWS accounts that you want to use as trusted signers, add the accounts to your distribution:

- **Download distributions:** Trusted signers are associated with cache behaviors. This allows you to require signed URLs for some objects and not for others in the same distribution. Trusted signers can only create signed URLs for objects that are associated with the corresponding cache behaviors. For example, if you have one trusted signer for one cache behavior and a different trusted signer for a different cache behavior, neither trusted signer can create signed URLs for objects associated with the other cache behavior.
- **Streaming distributions:** Trusted signers are associated with the distribution. After you add trusted signers to a streaming distribution, users must use signed URLs to access any of the objects associated with the distribution.

You must carefully define path patterns for your cache behaviors to ensure that CloudFront does or doesn't require signed URLs as appropriate.

> **Caution**
> Define path patterns and their sequence carefully to avoid providing end users with unintended access to your content. For example, suppose a request matches the path pattern for two cache behaviors. The first cache behavior does not require signed URLs and the second cache behavior does require signed URLs. End users will be able to access the objects without using a signed URL because CloudFront processes the cache behavior that is associated with the first match.

For more information about path patterns, see Path Pattern (p. 36).

> **Caution**
> If you're updating a distribution that you're already using to distribute content, add trusted signers only when you're ready to start generating signed URLs for your objects. After you add trusted signers to a streaming distribution, users must use signed URLs to access any of the objects associated with the distribution. After you add trusted signers to a cache behavior for a download distribution, users must use signed URLs to access the objects associated with the cache behavior.

The maximum number of trusted signers depends on the type of distribution:

- **Download distributions:** A maximum of five for each cache behavior
- **Streaming distributions:** A maximum of five for the distribution

You can add trusted signers to your distribution using either the CloudFront console or the CloudFront API. See the applicable topic:

- Adding Trusted Signers to Your Distribution Using the CloudFront Console (p. 99)
- Adding Trusted Signers to Your Distribution Using the CloudFront API (p. 100)

Adding Trusted Signers to Your Distribution Using the CloudFront Console

To add trusted signers to your distribution using the CloudFront console

1. If you want to use only the AWS account that created the distribution as a trusted signer, skip to Step 2.

If you want to use other AWS accounts, get the AWS account number for each account:

a. Sign in to the AWS Management Console at https://console.aws.amazon.com/console/home using an account that you want to use as a trusted signer.

b. In the upper-right corner of the console, click the name associated with the account, and click **My Account**.

c. Make note of the account number that appears in the upper-right corner of the page.

d. Click **Sign Out**.

e. Repeat Step a through Step d for the other accounts that you want to use as trusted signers.

2. Open the Amazon CloudFront console at https://console.aws.amazon.com/cloudfront/, and sign in using the account that you used to create the distribution to which you want to add trusted signers.

3. Click the icon for the distribution.

4. Change to edit mode:

    *   **Download distributions:** Click the **Behaviors** tab, click the behavior that you want to edit, and click **Edit**.
    *   **Streaming distributions:** Click **Edit**.

5. For **Restrict Viewer Access (Use Signed URLs)**, click **Yes**.

6. For **Trusted Signers**, check the applicable check boxes:

    *   **Self:** Check this check box if you want to use the current account (the account that you used to create the distribution).
    *   **Specify Accounts: Check this check box if you want to use other AWS accounts.**

7. If you checked the **Specify Accounts** check box, enter AWS account numbers (with or without hyphens) in the **AWS Account Number** field. Enter one account number per line.

8. Click **Yes, Edit**.

9. If you're adding trusted signers to a download distribution and you have more than one cache behavior, repeat Step 4 through Step 8 as applicable.

Adding Trusted Signers to Your Distribution Using the CloudFront API

You can use the CloudFront API to add the AWS account numbers for trusted signers to an existing distribution or to create a new distribution that includes trusted signers. In either case, specify the applicable values in the `TrustedSigners` element. For download distributions, add the `TrustedSigners` element to one or more cache behaviors. For streaming distributions, add the `TrustedSigners` element to the distribution.

See the applicable topic in the *Amazon CloudFront API Reference*:

*   **Create a new download distribution:** POST Distribution
*   **Update an existing download distribution:** PUT Distribution Config
*   **Create a new streaming distribution:** POST Streaming Distribution
*   **Update an existing streaming distribution:** PUT Streaming Distribution Config

Verifying that Trusted Signers Are Active (Optional)

After you add trusted signers to your distribution, you may want to verify that the signers are active. For a trusted signer to be active, the following must be true:

- The AWS account must have at least one active key pair. If you're rotating key pairs, the account will temporarily have two active key pairs, the old key pair and the new one.
- 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.

> **Note**
> To display a list of active trusted signers for a distribution, you currently must use the CloudFront API. A list of active trusted signers is not available in the CloudFront console.

Verifying that Trusted Signers Are Active Using the CloudFront API

To determine which trusted signers have active key pairs (are active trusted signers), you get the distribution and review the values in the `ActiveTrustedSigners` element. This element lists the AWS account number of each account that the distribution identifies as a trusted signer. If the trusted signer has one or more active CloudFront key pairs, the `ActiveTrustedSigners` element also lists the key pair IDs. For more information, see the applicable topic in the *Amazon CloudFront API Reference*:

- **Download distributions:** GET Distribution
- **Streaming distributions:** GET Streaming Distribution

Rotating CloudFront Key Pairs

AWS recommends that you rotate (change) your active CloudFront key pairs every 90 days. To rotate CloudFront key pairs that you are using to create signed URLs without invalidating URLs that haven't expired yet, perform the following tasks:

1. Create a new key pair for each of the accounts that you are using to create signed URLs.
2. Verify that CloudFront is aware of the new key pairs. For more information, see Verifying that Trusted Signers Are Active (Optional) (p. 101).
3. Update your application to create signatures using the private keys from the new key pairs.
4. Confirm that signed URLs that you are signing using the new private keys are working.
5. Wait until the expiration date has passed in signed URLs that were signed using the old CloudFront key pairs.
6. Change the old CloudFront key pairs to Inactive.
7. Reconfirm that signed URLs that you are signing using the new private keys are working.
8. In the AWS Management Console, delete the old CloudFront key pairs.
9. Delete the old private keys from the location where you stored them.

For more information, go to Access Credential Rotation on the About AWS Security Credentials page.

Overview of Signed URLs

Topics

Now that you've configured CloudFront to require that end users access your objects using special signed URLs, you can create the signed URLs (either manually or programmatically) and distribute them to the users that you want to have access to the objects.

A signed URL includes additional information, for example, an expiration date and time, that gives you more control over access to your content. This additional information appears in a policy statement, which is based on either a canned policy or a custom policy. The differences between canned and custom policies are explained in the next two sections.

> **Note**
> You can create some signed URLs using canned policies and create some signed URLs using custom policies for the same distribution.

Signed URLs that You Create Using a Canned Policy

Use a canned policy to create a CloudFront signed URL if you want to:

- Restrict access to a single object.
- Control access to your objects based only on the date and time that you want users to stop having access.

You don't include the policy in the URL, so a canned policy results in a shorter URL. You do include a signature, for which you hash and sign the policy.

When a user requests an object using a signed URL that you created using a canned policy, CloudFront reconstructs the canned policy statement based on information in the URL. CloudFront then compares the reconstructed policy statement with the policy statement in the signature to determine whether to allow the end user to access to the content. If the two statements don't match exactly, CloudFront denies access to the content.

For information about creating signed URLs using a canned policy, see Creating a Signed URL Using a Canned Policy (p. 106).

Signed URLs that You Create Using a Custom Policy

Use a custom policy to create a CloudFront signed URL if you want to:

- Restrict access to one or more objects, for example, all of the .pdf files in an annual-report directory. This allows you to use a single policy statement for the signed URLs for multiple objects.
- Control access to your objects based on:
  - The date and time that you want users to stop having access.

- The date and time that you want users to begin having access. (Optional)
- The IP address or range of IP addresses of the users who you want to have access to the object or objects. (Optional)

For a signed URL that you create using a custom policy, you hash and sign the policy and include the signature, as you do with a signed URL that you create using a canned policy. In addition, you include a Base64-encoded version of the policy in the URL, which provides URL-safe compression. As a result, using a custom policy results in a longer URL than a canned policy.

When a user requests an object using a signed URL that you created using a canned policy, CloudFront compares the custom policy statement in the signed URL with the policy statement in the signature to determine whether to allow the end user to access to the content. If the two statements don't match exactly, CloudFront denies access to the content.

For information about creating signed URLs using a custom policy, see Creating a Signed URL Using a Custom Policy (p. 111).

The Parts of a Signed URL

CloudFront signed URLs include the following components.

Base URL

The base URL is the CloudFront URL that you would use to access the object if you were not using signed URLs, including your own query string parameters, if any. For more information about the format of URLs for download distributions, see Format of URLs for CloudFront Objects (p. 58).

The following examples show values that you specify for download distributions.

- The following CloudFront URL is for an object in a download distribution (using the CloudFront domain name). Note that `image.jpg` is in an `images` directory. The path to the object in the URL must match the path to the object on your HTTP server or in your Amazon S3 bucket.

  `http://d111111abcdef8.cloudfront.net/images/image.jpg`
- The following CloudFront URL includes a query string:

  `http://d111111abcdef8.cloudfront.net/images/image.jpg?size=large`
- The following CloudFront URLs are for objects in a download distribution. Both use an alternate domain name; the second one includes a query string:

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

  `http://www.example.com/images/image.jpg?color=red`
- The following CloudFront URL is for an objects in a download distribution that uses an alternate domain name and the HTTPS protocol:

  `https://www.example.com/images/image.jpg`

For streaming distributions, the following examples are for objects in two different video formats, MP4 and FLV:

- **MP4:** `mp4:sydney-vacation.mp4`
- **FLV:** `sydney-vacation`
- **FLV:** `sydney-vacation.flv`

Note
For .flv files, whether you include the `.flv` filename extension depends on your player. 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, sydney-vacation instead of sydney-vacation.mp4).

Expiration Date and Time (Canned Policies Only)

The date and time, in Unix time format (in seconds) and Coordinated Universal Time (UTC), that you want the URL to stop allowing access to the object. For example, January 1, 2013 10:00 am UTC converts to 1357034400 in Unix time format. For information about UTC, see *RFC 3339, Date and Time on the Internet: Timestamps*, http://tools.ietf.org/html/rfc3339.

If you're using a custom policy, you can still specify the date and time that you want a signed URL to stop allowing access to the object, but the date and time is included in the policy statement instead of appearing as a separate query string parameter in the signed URL.

Policy Statement

A policy statement is a text string in JSON format that determines the characteristics of the signed URL. For both canned policies and custom policies, the policy statement includes the URL of the object (for download distributions) or the stream name (for streaming distributions), and an expiration date and time. Note the following:

- **Canned policy:** You don't include the policy statement in the signed URL—you only create a policy statement so you can hash and sign it, and include the signature in the URL. See Signature (p. 104).
- **Custom policy:** You can also optionally include a date and time that the URL becomes valid and/or an IP address or range of IP addresses that are allowed to access the object. Then you Base64-encode the policy statement, and you include the encoded policy statement in the signed URL.

Signature

The signature is a hashed and signed version of the policy statement.

CloudFront Key Pair ID for the AWS Account that Is Creating the Signed URL

The CloudFront key pair ID tells CloudFront which public key to use to validate the signed URL. CloudFront compares the information in the signature with the information in the policy statement to verify that the URL has not been tampered with.

The key pair ID that you include in CloudFront signed URLs must be the ID of an active key pair for one of your trusted signers. For more information, see Specifying the AWS Accounts That Can Create Signed URLs (Trusted Signers) (p. 97).

If you make a key pair inactive while rotating CloudFront key pairs, and if you're generating signed URLs programmatically, you must update your application to use a new active key pair for one of your trusted signers. If you're generating signed URLs manually, you must create new signed URLs. For more information about rotating key pairs, see Rotating CloudFront Key Pairs (p. 101).

Amazon CloudFront Developer Guide
When Does CloudFront Check the Expiration Date and
Time in a Signed URL?

When Does CloudFront Check the Expiration Date and Time in a Signed URL?

When CloudFront checks the expiration date and time in a signed URL to determine whether the URL is still valid depends on whether the URL is for a download distribution or a streaming distribution:

- **Download distributions:** CloudFront checks the expiration date and time in a signed URL at the time of the HTTP request. If a client begins to download a large object immediately before the expiration time, the download should complete even if the expiration time passes during the download. If the TCP connection drops and the client tries to restart the download after the expiration time passes, the download will fail.

  If a client uses Range GETs to get an object in smaller pieces, any GET request that occurs after the expiration time passes will fail. For more information about Range GETs, see How CloudFront Processes Partial Requests for an Object (Range GETs) (p. 72).

- **Streaming distributions:** CloudFront checks the expiration time in a signed URL at the start of a play event. If a client starts to play a media file before the expiration time passes, CloudFront allows the entire media file to play. However, depending on the media player, pausing and restarting might trigger another play event. Skipping to another position in the media file will trigger another play event. If the subsequent play event occurs after the expiration time passes, CloudFront won't serve the media file.

Creating a Signed URL Using a Canned Policy

To create a signed URL using a canned policy

1. If you're using .NET or Java to create signed URLs, and if you haven't reformatted the private key for your key pair from the default .pem format to a format compatible with .NET or with Java, do so now. For more information, see Reformatting the CloudFront Private Key (.NET and Java Only) (p. 98).
2. Concatenate the following values in the specified order, and remove the whitespace between the parts. You might have to include escape characters in the string in application code. All values have a type of String. Each part is keyed by number (**1**) to the two examples that follow.

1	*Base URL for the object* This is the URL that you use to access the object if you aren't using a signed URL, for example: • **Download distribution:** `http://d111111abcdef8.cloudfront.net/images/image.jpg` • **Streaming distribution:** `videos/mediafile.flv`
2	**?** The **?** indicates that query string parameters follow the base URL. Include the **?** even if you don't have any query string parameters of your own.

3	*Your query string parameters, if any***&**
	
	 This value is optional. If you want to add your own query string parameters, for example:
	
	 `color=red&size=medium`
	
	 then add the parameters after the **?** (see **2**) and before the `Expires` parameter.
	
	     Important
	    Your parameters cannot be named `Expires`, `Signature`, or `Key-Pair-Id`.
	
	 If you add your own parameters, append an **&** after each one, including the last one.

4	**Expires=***date and time in Unix time format (in seconds) and Coordinated Universal Time (UTC)* Specify the expiration date and time in Unix time format and Coordinated Universal Time (UTC). For example, January 1, 2013 10:00 am UTC converts to 1357034400 in Unix time format. For information about UTC, see *RFC 3339, Date and Time on the Internet: Timestamps*, http://tools.ietf.org/html/rfc3339.
5	**&Signature=***hashed and signed version of the policy statement* A hashed and signed version of the policy statement. For more information, see Creating a Signature for a Canned Policy (p. 107).

⑥ **&Key-Pair-Id=**_active CloudFront key pair Id for the key pair that you are using to generate the signature_

The ID for an active CloudFront key pair, for example, APKA9ONS7QCOWEXAMPLE:

• **Download distributions:** The key pair must be associated with an AWS account that is one of the trusted signers for the applicable cache behavior.

• **Streaming distributions:** The key pair must be associated with an AWS account that is one of the trusted signers for the distribution.

For more information, see Specifying the AWS Accounts That Can Create Signed URLs (Trusted Signers) (p. 97).

Example signed URL for a download distribution:

① **http://d111111abcdef8.cloudfront.net/image.jpg** ②**?** ③**color=red&size=medium&**
④**Expires=1357034400** ⑤**&Signature=nitfHRCrtziwO2HwPfWw~yYDhUF5EwRunQA-j19DzZr vDh6hQ73lDx~-ar3UocvvRQVw6EkC~GdpGQyyOSKQim-TxAnW7d8F5Kkai9HVx0Flu-5jcQb0UEmatEXAMPLE3ReXySpLSMj0yCd3ZAB4UcBCAqEijkytL6f3fVYNGQI6**
⑥**&Key-Pair-Id=APKA9ONS7QCOWEXAMPLE**

Example signed URL for a streaming distribution:

① **videos/mediafile.flv** ②**?** ③**color=red&size=medium&** ④**Expires=1357034400**
⑤**&Signature=nitfHRCrtziwO2HwPfWw~yYDhUF5EwRunQA-j19DzZr vDh6hQ73lDx~-ar3UocvvRQVw6EkC~GdpGQyyOSKQim-TxAnW7d8F5Kkai9HVx0Flu-5jcQb0UEmatEXAMPLE3ReXySpLSMj0yCd3ZAB4UcBCAqEijkytL6f3fVYNGQI6**
⑥**&Key-Pair-Id=APKA9ONS7QCOWEXAMPLE**

Creating a Signature for a Canned Policy

To create the signature for a signed URL that uses a canned policy, you perform two procedures:

• With the first procedure, immediately following, you create a policy statement.

• With the second procedure, you hash and sign the policy statement. There are two versions of this procedure. The version that you choose depends on your distribution type (download or streaming) and, for streaming distributions, the media player that you're using (Adobe Flash Player or another media player). Use the links after the first procedure to guide you to the applicable version of the second procedure.

For signed URLs that use a canned policy, you don't include the policy statement in the URL, as you do for signed URLs that use a custom policy.

For additional information and examples of how to hash, sign, and encode the policy statement, see:

• Using Linux Commands and OpenSSL for Base64-Encoding and Encryption (p. 119)
• Code and Examples for Creating a Signature for a Signed URL (p. 121)
• Tools for Configuring Private Content (p. 254)

To create the policy statement for a signed URL that uses a canned policy

1. Construct the policy statement using the following JSON format and using UTF-8 character encoding. Include all punctuation and other literal values exactly as specified.

   ```
   {"Statement":[{"Resource":"base URL or stream name",
   "Condition":{"DateLessThan":{"AWS:EpochTime":ending date and time in Unix
   time format and UTC}}}]}
   ```

   For `Resource` and `DateLessThan`, specify the following values:

   Resource: The value that you specify depends on whether you're creating the signed URL for a download distribution or a streaming distribution:

   - **Download distributions:** The base URL including your query strings, if any, but excluding the CloudFront `Expires`, `Signature`, and `Key-Pair-Id` parameters, for example:

     ```
     http://d111111abcdef8.cloudfront.net/images/horizon.jpg?size=large&license=yes
     ```

     Note the following:
     - The value must begin with `http://`, `https://`, or `http*://`.
     - If you have no query string parameters, omit the question mark.
     - You can use the wild card character that matches zero or more characters (*) or the wild-card character that matches exactly one character (?) anywhere in the string. For example, the value:

       ```
       http*://d111111abcdef8.cloudfront.net/*game_download.zip*
       ```

       would include (for example), the following objects:

       ```
       http://d111111abcdef8.cloudfront.net/example_game_download.zip?license=yes
       ```

       ```
       https://d111111abcdef8.cloudfront.net/example_game_download.zip?license=yes
       ```

       ```
       http://d111111abcdef8.cloudfront.net/test_game_download.zip?license=temp
       ```

       ```
       https://d111111abcdef8.cloudfront.net/test_game_download.zip?license=temp
       ```
     - If you specify an alternate domain name (CNAME) in the URL, you must specify the alternate domain name when referencing the object in your web page or application. Do not specify the Amazon S3 URL for the object.

   - **Streaming distributions:** Include only the stream name. For example, if the full URL for a streaming video is:

     ```
     rtmp://s5c39gqb8ow64r.cloudfront.net/videos/mp3_name.mp3
     ```

     then use the following value for `Resource`:

     ```
     videos/mp3_name
     ```

     Do not include a prefix such as `mp3:` or `mp4:`. Also, depending on the player you're using, you might have to omit the file extension from the value of `Resource`. For example, you might need to use `sydney-vacation` instead of `sydney-vacation.flv`.

   DateLessThan: The expiration date and time for the URL in Unix time format (in seconds) and Coordinated Universal Time (UTC). For example, January 1, 2013 10:00 am UTC converts to 1357034400 in Unix time format. For information about UTC, see *RFC 3339, Date and Time on the Internet: Timestamps*, http://tools.ietf.org/html/rfc3339.

This value must match the value of the `Expires` query string parameter in the signed URL. Do not enclose the value in quotation marks.

For more information, see When Does CloudFront Check the Expiration Date and Time in a Signed URL? (p. 105).

Example

When you use the following sample canned policy in a signed URL, an end user can access the object `http://d111111abcdef8.cloudfront.net/horizon.jpg` until January 1, 2013 10:00 am UTC:

```
{"Statement":[{"Resource":"http://d111111abcdef8.cloudfront.net/horizon.jpg?size=large&license=yes","Condition":{"DateLessThan":{"AWS:EpochTime":1357034400}}}]}
```

If you copy and paste this example, replace the URL and expiration time with your own values.

2. Remove any whitespace from the policy statement. You might have to include escape characters in the string in application code.

Perform the applicable procedure to create the signature for your signed URL:

*   Option 1: To create a signature for a download distribution or for a streaming distribution (without Adobe Flash Player) by using a canned policy (p. 109)
*   Option 2: To create a signature for a streaming distribution by using a canned policy (Adobe Flash Player) (p. 110)

Option 1: To create a signature for a download distribution or for a streaming distribution (without Adobe Flash Player) by using a canned policy

1. Use the SHA-1 hash function to hash and sign the policy statement that you created in the To create the policy statement for a signed URL that uses a canned policy (p. 108) procedure. For the private key that is required by the hash function, use the private key that is associated with the applicable active trusted signer.

    Note
    The method that you use to hash and sign the policy statement depends on your programming language and platform. For sample code, see Code and Examples for Creating a Signature for a Signed URL (p. 121).

2. Remove whitespace from the hashed and signed string.
3. Base64-encode the string.
4. Replace characters that are invalid in a URL query string with characters that are valid. The following table lists invalid and valid characters.

Replace these invalid characters	With these valid characters
+	- (hyphen)
=	_ (underscore)
/	~ (tilde)

5. Append the resulting value to your signed URL after **&Signature=**, and return to To create a signed URL using a canned policy (p. 106) to finish concatenating the parts of your signed URL.

Option 2: To create a signature for a streaming distribution by using a canned policy (Adobe Flash Player)

1. Use the SHA-1 hash function to hash and sign the policy statement that you created in the To create the policy statement for a signed URL that uses a canned policy (p. 108) procedure. For the private key that is required by the hash function, use the private key that is associated with the applicable active trusted signer.

    Note
    The method that you use to hash and sign the policy statement depends on your programming language and platform. For sample code, see Code and Examples for Creating a Signature for a Signed URL (p. 121).

2. Remove whitespace from the hashed and signed string.

    Continue on to Step 3 if you're using Adobe Flash Player and the stream name is passed in from a web page.

    If you're using Adobe Flash Player and if the stream name is not passed in from a web page, skip the rest of this procedure. For example, if you wrote your own player that fetches stream names from within the Adobe Flash .swf file, skip the rest of this procedure.

3. Base64-encode the string.

4. Replace characters that are invalid in a URL query string with characters that are valid. The following table lists invalid and valid characters.

Replace these invalid characters	With these valid characters
+	- (hyphen)
=	_ (underscore)
/	~ (tilde)

5. Some versions of Adobe Flash Player require that you URL-encode the characters ?, =, and &. For information about whether your version of Adobe Flash Player requires this character substitution, refer to the Adobe website.

    If your version of Flash does not require URL-encoding those character, skip to Step 6.

    If your version of Flash requires URL-encoding those characters, replace them as indicated in the following table. (You already replaced = in the previous step.)

Replace these invalid characters	With this URL encoding
?	%3F
&	%26

6. Append the resulting value to your signed URL after **&Signature=**, and return to To create a signed URL using a canned policy (p. 106) to finish concatenating the parts of your signed URL.

Creating a Signed URL Using a Custom Policy

Topics

To create a signed URL using a custom policy, perform the following procedure.

To create a signed URL using a custom policy

1. If you're using .NET or Java to create signed URLs, and if you haven't reformatted the private key for your key pair from the default .pem format to a format compatible with .NET or with Java, do so now. For more information, see Reformatting the CloudFront Private Key (.NET and Java Only) (p. 98).

2. Concatenate the following values in the specified order, and remove the whitespace between the parts. You might have to include escape characters in the string in application code. All values have a type of String. Each part is keyed by number (**1**) to the two examples that follow.

1	*Base URL for the object* This is the URL that you use to access the object if you aren't using a signed URL, for example: • **Download distribution:** `http://d111111abcdef8.cloudfront.net/images/image.jpg` • **Streaming distribution:** `videos/mediafile.flv`
2	**?** The **?** indicates that query string parameters follow the base URL. Include the **?** even if you don't have any query string parameters of your own.

3	*Your query string parameters, if any***&**
	
	This value is optional. If you want to add your own query string parameters, for example:
	
	`color=red&size=medium`
	
	then add them after the **?** (see **2**) and before the `Policy` parameter.
	
	    Important
	    Your parameters cannot be named `Policy`, `Signature`, or `Key-Pair-Id`.
	
	If you add your own parameters, append an **&** after each one, including the last one.

4	**Policy=***policy statement* Your policy statement in JSON format, with white space removed. For more information, see Creating a Policy Statement for a Custom Policy (p. 112).
5	**&Signature=***hashed and signed version of the policy statement* A hashed and signed version of the policy statement. For more information, see Creating a Signature for a Custom Policy (p. 116).

> ⑥ **&Key-Pair-Id=**_active CloudFront key pair Id for the key pair that you are using to sign the policy statement_
>
> The ID for an active CloudFront key pair, for example, APKA9ONS7QCOWEXAMPLE:
>
> - **Download distributions:** The key pair must be associated with an AWS account that is one of the trusted signers for the applicable cache behavior.
> - **Streaming distributions:** The key pair must be associated with an AWS account that is one of the trusted signers for the distribution.
>
> For more information, see Specifying the AWS Accounts That Can Create Signed URLs (Trusted Signers) (p. 97).

Example signed URL for a download distribution:

①**http://d111111abcdef8.cloudfront.net/image.jpg** ②**?** ③**color=red&size=medium&**

④**Policy=eyANCiAgICEXAMPLEW1lbnQiOiBbeyANCiAgICAgICJSZXNvdXJjZSI6Imh0dHA 6Ly9kemJlc3FtN3VuMW0wLmNsb3VkZnJvbnQubmV0L2RlbW8ucGhwIiwgDQogICAgICAiQ 29uZGl0aW9uIjp7IA0KICAgICAgICAgIklwQWRkcmVzcyI6eyJBV1M6U291cmNlSXAiOiI yMDcuMTcxLjE4MC4xMDEvMzIifSwNCiAgICAgICAgICJEYXRRIR3JIYXRlclRoYW4iOnsiQ VdTOkVwb2NoVGltZSI6MTI5Njg2MDE3Nn0sDQogICAgICAgICAiRGF0ZUxlc3NUaGFuIjp 7IkFXUzpFcG9jG9jaFRpbWUiOjEyOTY4NjAyMjZ9DQogICAgICB9IA0KICAgfV0gDQp9DQo**

⑤**&Signature=nitfHRCrtziwO2HwPfWw~yYDhUF5EwRunQA-j19DzZrvDh6hQ73lDx~ -ar3UocvvRQVw6EkC~GdpGQyyOSKQim-TxAnW7d8F5Kkai9HVx0Flu-5jcQb0UEmat EXAMPLE3ReXySpLSMj0yCd3ZAB4UcBCAqEijkytL6f3fVYNGQI6**

⑥**&Key-Pair-Id=APKA9ONS7QCOWEXAMPLE**

Example signed URL for a streaming distribution:

①**videos/mediafile.flv** ②**?** ③**color=red&size=medium&**

④**Policy=eyANCiAgICEXAMPLEW1lbnQiOiBbeyANCiAgICAgICJSZXNvdXJjZSI6Imh0dHA 6Ly9kemJlc3FtN3VuMW0wLmNsb3VkZnJvbnQubmV0L2RlbW8ucGhwIiwgDQogICAgICAiQ 29uZGl0aW9uIjp7IA0KICAgICAgICAgIklwQWRkcmVzcyI6eyJBV1M6U291cmNlSXAiOiI yMDcuMTcxLjE4MC4xMDEvMzIifSwNCiAgICAgICAgICJEYXRRIR3JIYXRlclRoYW4iOnsiQ VdTOkVwb2NoVGltZSI6MTI5Njg2MDE3Nn0sDQogICAgICAgICAiRGF0ZUxlc3NUaGFuIjp 7IkFXUzpFcG9jG9jaFRpbWUiOjEyOTY4NjAyMjZ9DQogICAgICB9IA0KICAgfV0gDQp9DQo**

⑤**&Signature=nitfHRCrtziwO2HwPfWw~yYDhUF5EwRunQA-j19DzZrvDh6hQ73lDx~ -ar3UocvvRQVw6EkC~GdpGQyyOSKQim-TxAnW7d8F5Kkai9HVx0Flu-5jcQb0UEmat EXAMPLE3ReXySpLSMj0yCd3ZAB4UcBCAqEijkytL6f3fVYNGQI6**

⑥**&Key-Pair-Id=APKA9ONS7QCOWEXAMPLE**

Creating a Policy Statement for a Custom Policy

To create a policy statement for a custom policy, perform the following procedure. For several example policy statements that control access to objects in a variety of ways, see Example Policy Statements for a Custom Policy (p. 114).

To create the policy statement for a signed URL that uses a custom policy

1.   Construct the policy statement using the following JSON format.

```
{
    "Statement": [{
        "Resource":"URL or stream name of the object",
        "Condition":{
            "DateLessThan":{"AWS:EpochTime":required ending date and time in
Unix time format and UTC},
            "DateGreaterThan":{"AWS:EpochTime":optional beginning date and time
 in Unix time format and UTC},
            "IpAddress":{"AWS:SourceIp":"optional IP address"}
        }
    }]
}
```

Note the following:

- Use UTF-8 character encoding.
- Include all punctuation and parameter names exactly as specified. Abbreviations for parameter names are not accepted.
- The order of the parameters in the `Condition` section doesn't matter.
- For information about the values for `Resource`, `DateLessThan`, `DateGreaterThan`, and `IpAddress`, see the descriptions after this procedure.

2. Remove any whitespace from the policy statement. You might have to include escape characters in the string in application code.

3. Append the resulting value to your signed URL after `Policy=`.

4. Create a signature for the signed URL by hashing, signing, and Base64-encoding the policy statement. For more information, see Creating a Signature for a Custom Policy (p. 116).

Resource

- **Download distributions (optional but recommended):** The base URL including your query strings, if any, but excluding the CloudFront `Policy`, `Signature`, and `Key-Pair-Id` parameters, for example:

```
http://d111111abcdef8.cloudfront.net/images/horizon.jpg?size=large&license=yes
```

> **Caution**
> If you omit the Resource parameter for a download distribution, end users can access all of the objects associated with any distribution that is associated with the key pair that you use to create the signed URL.

Note the following:

- The value must begin with `http://`, `https://`, or *.
- If you have no query string parameters, omit the question mark.
- You can use the wild card character that matches zero or more characters (*) or the wild-card character that matches exactly one character (?) anywhere in the string. For example, the value:

```
http*://d111111abcdef8.cloudfront.net/*game_download.zip*
```

would include (for example), the following objects:

```
http://d111111abcdef8.cloudfront.net/example_game_download.zip?license=yes
```

```
https://d111111abcdef8.cloudfront.net/example_game_download.zip?license=yes
```

```
http://d111111abcdef8.cloudfront.net/test_game_download.zip?license=temp

https://d111111abcdef8.cloudfront.net/test_game_download.zip?license=temp
```

- If you specify an alternate domain name (CNAME) in the URL, you must specify the alternate domain name when referencing the object in your web page or application. Do not specify the Amazon S3 URL for the object.

- **Streaming distributions:** Include only the stream name. For example, if the full URL for a streaming video is:

```
rtmp://s5c39gqb8ow64r.cloudfront.net/videos/mp3_name.mp3
```

then use the following value for `Resource`:

```
videos/mp3_name
```

Do not include a prefix such as `mp3:` or `mp4:`. Also, depending on the player you're using, you might have to omit the file extension from the value of `Resource`. For example, you might need to use `sydney-vacation` instead of `sydney-vacation.flv`.

DateLessThan

The expiration date and time for the URL in Unix time format (in seconds) and Coordinated Universal Time (UTC). Do not enclose the value in quotation marks. For information about UTC, see *RFC 3339, Date and Time on the Internet: Timestamps*, http://tools.ietf.org/html/rfc3339.

For example, January 1, 2013 10:00 am UTC converts to 1357034400 in Unix time format.

This is the only required parameter in the `Condition` section. CloudFront requires this value to prevent users from having permanent access to your private content.

For more information, see When Does CloudFront Check the Expiration Date and Time in a Signed URL? (p. 105)

DateGreaterThan (Optional)

An optional start date and time for the URL in Unix time format (in seconds) and Coordinated Universal Time (UTC). Users are not allowed to access the object before the specified date and time. Do not enclose the value in quotation marks.

IpAddress (Optional)

The IP address of the client making the GET request. To allow any IP address to access the object, omit this parameter.

IP address ranges must be in standard IPv4 CIDR format (for example, `10.52.176.0/24`). For more information, go to *RFC 4632, Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan*, http://tools.ietf.org/html/rfc4632.

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 separate ranges.

Example Policy Statements for a Custom Policy

The following example policy statements show how to control access to a specific object, all of the objects in a directory, or all of the objects associated with a key pair ID. The examples also show how to control

access from an individual IP address or a range of IP addresses, and how to prevent users from using the signed URL after a specified date and time.

If you copy and paste any of these examples, remove any whitespace, replace the applicable values with your own values, and include a newline character after the closing brace (}).

Example Policy Statement: Accessing One Object from a Range of IP Addresses

The following example custom policy in a signed URL specifies that an end user can access the object `http://d111111abcdef8.cloudfront.net/game_download.zip` from IP addresses in the range `192.0.2.0/24` until January 1, 2013 10:00 am UTC:

```
{
   "Statement": [{
      "Resource":"http://d111111abcdef8.cloudfront.net/game_download.zip",
      "Condition":{
         "IpAddress":{"AWS:SourceIp":"192.0.2.0/24"},
         "DateLessThan":{"AWS:EpochTime":1357034400}
      }
   }]
}
```

Example Policy Statement: Accessing All Objects in a Directory from a Range of IP Addresses

The following example custom policy allows you to create signed URLs for any object in the `training` directory, as indicated by the * wildcard character in the `Resource` parameter. End users can access the object from an IP address in the range `192.0.2.0/24` until January 1, 2013 10:00 am UTC:

```
{
   "Statement": [{
      "Resource":"http://d111111abcdef8.cloudfront.net/training/*",
      "Condition":{
         "IpAddress":{"AWS:SourceIp":"192.0.2.0/24"},
         "DateLessThan":{"AWS:EpochTime":1357034400}
      }
   }]
}
```

Each signed URL in which you use this policy includes a base URL that identifies a specific object, for example:

`http://d111111abcdef8.cloudfront.net/training/orientation.pdf`

Example Policy Statement: Accessing All Objects Associated with a Key Pair ID from One IP Address

The following sample custom policy allows you to create signed URLs for any object associated with any distribution, as indicated by the * wildcard character in the `Resource` parameter. The end user must use the IP address `192.0.2.10/32`. (The value `192.0.2.10/32` in CIDR notation refers to a single IP address, `192.0.2.10`.) The objects are available only from January 1, 2013 10:00 am UTC until January 2, 2013 10:00 am UTC:

```
{
   "Statement": [{
      "Resource":"http://*",
      "Condition":{
         "IpAddress":{"AWS:SourceIp":"192.0.2.10/32"},
         "DateGreaterThan":{"AWS:EpochTime":1357034400},
         "DateLessThan":{"AWS:EpochTime":1357120800}
      }
   }]
}
```

Each signed URL in which you use this policy includes a base URL that identifies a specific object in a specific CloudFront distribution, for example:

```
http://d111111abcdef8.cloudfront.net/training/orientation.pdf
```

The signed URL also includes a key pair ID, which must be associated with a trusted signer in the distribution (d111111abcdef8.cloudfront.net) that you specify in the base URL.

Creating a Signature for a Custom Policy

The signature for a signed URL that uses a custom policy is a hashed, signed, and Base64-encoded version of the policy statement. To create a signature for a custom policy, perform the applicable procedure. The version that you choose depends on your distribution type (download or streaming) and, for streaming distributions, the media player that you're using (Adobe Flash Player or another media player):

- Option 1: To create a signature for a download distribution or for a streaming distribution (without Adobe Flash Player) by using a custom policy (p. 116)
- Option 2: To create a signature for a streaming distribution by using a custom policy (Adobe Flash Player) (p. 117)

For additional information and examples of how to hash, sign, and encode the policy statement, see:

- Using Linux Commands and OpenSSL for Base64-Encoding and Encryption (p. 119)
- Code and Examples for Creating a Signature for a Signed URL (p. 121)
- Tools for Configuring Private Content (p. 254)

Option 1: To create a signature for a download distribution or for a streaming distribution (without Adobe Flash Player) by using a custom policy

1. Use the SHA-1 hash function to hash and sign the policy statement that you created in the To create the policy statement for a signed URL that uses a custom policy (p. 112) procedure. For the private key that is required by the hash function, use the private key that is associated with the applicable active trusted signer.

   Note
   The method that you use to hash and sign the policy statement depends on your programming language and platform. For sample code, see Code and Examples for Creating a Signature for a Signed URL (p. 121).

2. Remove whitespace from the hashed and signed string.
3. Base64-encode the string.
4. Replace characters that are invalid in a URL query string with characters that are valid. The following table lists invalid and valid characters.

Replace these invalid characters	With these valid characters
+	- (hyphen)
=	_ (underscore)
/	~ (tilde)

5. Append the resulting value to your signed URL after **&Signature=**, and return to To create a signed URL using a custom policy (p. 111) to finish concatenating the parts of your signed URL.

Option 2: To create a signature for a streaming distribution by using a custom policy (Adobe Flash Player)

1. Use the SHA-1 hash function to hash and sign the policy statement that you created in the To create the policy statement for a signed URL that uses a custom policy (p. 112) procedure. For the private key that is required by the hash function, use the private key that is associated with the applicable active trusted signer.

    Note
    The method that you use to hash and sign the policy statement depends on your programming language and platform. For sample code, see Code and Examples for Creating a Signature for a Signed URL (p. 121).

2. Remove whitespace from the hashed and signed string.

    Continue on to Step 3 if the stream name is passed in from a web page.

    If the stream name is not passed in from a web page, skip the rest of this procedure. For example, if you wrote your own player that fetches stream names from within the Adobe Flash .swf file, skip the rest of this procedure.

3. Base64-encode the string.

4. Replace characters that are invalid in a URL query string with characters that are valid. The following table lists invalid and valid characters.

Replace these invalid characters	With these valid characters
+	- (hyphen)
=	_ (underscore)
/	~ (tilde)

5. Some versions of Adobe Flash Player require that you URL-encode the characters ?, =, and &. For information about whether your version of Adobe Flash Player requires this character substitution, refer to the Adobe website.

    If your version of Adobe Flash Player does not require that you URL-encode the characters ?, =, and &, skip to Step 6.

    If your version of Adobe Flash Player requires URL-encoding those characters, replace them as indicated in the following table. (You already replaced = in the previous step.)

Replace these invalid characters	With this URL encoding
?	%3F

Replace these invalid characters	With this URL encoding
&	%26

6. Append the resulting value to your signed URL after **&Signature=**, and return to To create a signed URL using a custom policy (p. 111) to finish concatenating the parts of your signed URL.

Using Linux Commands and OpenSSL for Base64-Encoding and Encryption

You can use Linux command-line commands and OpenSSL to:

* Base64-encode the policy statement and replace invalid characters with valid characters.
* Convert the policy statement into a signature.

For information about OpenSSL, go to http://www.openssl.org.

Base64-Encoding the Policy Statement

The following Linux command Base64-encodes the policy statement (in the file policy) and replaces characters that are not valid in URL query string parameters with characters that are valid:

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

where:

	`cat` sends the policy file to openssl.
	OpenSSL Base64-encodes the file.
	`tr` replaces characters that are not valid in URL query string parameters with characters that are valid.

Converting the Policy Statement into a Signature

The following Linux command hashes, signs, and Base64-encodes the policy statement to create a signature:

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

where:

	`cat` sends the Base64-encoded `policy` file to OpenSSL.
	OpenSSL hashes the file using SHA-1 and signs it using the private key file `private-key.pem`.
	OpenSSL Base64-encodes the hashed and signed policy statement.
	`tr` replaces characters that are not valid in URL query string parameters with characters that are valid.

Note
Remove whitespace, if any, from the resulting signature.

For code examples that demonstrate creating a signature in several programming languages see Code and Examples for Creating a Signature for a Signed URL (p. 121).

Code and Examples for Creating a Signature for a Signed URL

This section includes downloadable application samples that demonstrate how to create signatures for signed URLs. Examples are available in Perl, PHP, C#, and Java. 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 Private Content Works (p. 88).

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.

The following is a sample Base64-encoded stream name.

```
mp4:example/video.mp4%3FPolicy%3DewogICJTdGF0ZW1lbnQiOlt7CiAgICAgICJSZXNvdXJjZSI
6ImRyciIsCiAgICAgICJDb25kaXRpb24iOnsKICAgICAgICAiSXBBZGRyZXNzIjp7IkFXUzpTb3VyY2V
JcCI6IjAuMC4wLjAvMCJ9LAogICAgICAgICJEYXRlTGVzc1RoYW4iOnsiQVdTOkVwb2NoVGltZSI6MjE
0NTkxNjgwMH0KICAgICAgfQogICAgfQ1dfQ000_%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%3DewogICJTdGF0ZW1lbnQiQi

    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=ewogICJTdGF0ZW1lbnQiOiOlt7CiAgICAgICJSZXNvdXJjZSI6ImR
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 Private Content Works (p. 88).

In the following code segment, the function `rsa_sha1_sign` hashes and signs the policy statement. 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 Hashing in PHP

```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 .

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 Creating a Signed URL Using a Custom Policy (p. 111).

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 the OpenSSL implementation of SHA-1, see The Open Source Toolkit for SSL/TLS.

See also

- Tools for Configuring Private Content (p. 254)

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 Private Content Works (p. 88).

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 signing 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 and signed 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#

```
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 PrivateKey.xml, which you created by converting your
        // .pem file to the XML format that the .NET framework uses.
        // Several tools are available. We used
        // .NET 2.0 OpenSSL Public and Private Key Parser,
        // http://www.jensign.com/opensslkey/opensslkey.cs.
        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#

```csharp
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-pathToPolicyStmnt, 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 PrivateKey.xml, which you created by converting your
        // .pem file to the XML format that the .NET framework uses.
        // Several tools are available. We used
        // .NET 2.0 OpenSSL Public and Private Key Parser,
        // http://www.jensign.com/opensslkey/opensslkey.cs.
```

```
        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 Perl (p. 121)
- Create a URL Signature Using PHP (p. 122)
- Create a URL Signature Using Java (p. 133)
- Tools for Configuring Private Content (p. 254)

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 Private Content Works (p. 88).

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. 121)
- Create a URL Signature Using PHP (p. 122)
- Create a URL Signature Using C# and the .NET Framework (p. 125)
- Tools for Configuring Private Content (p. 254)

Using an HTTPS Connection to Access Your Objects

Topics

For download distributions, you can use HTTPS requests to ensure that your objects are encrypted when CloudFront serves them to viewers and, if you want, when CloudFront gets them from your origin. When you create a download distribution, you can:

- Configure one or more CloudFront cache behaviors to require that viewers use only the HTTPS protocol to access your objects in the CloudFront cache. This allows you to require HTTPS for some objects but not for others.
- Configure one or more CloudFront origins to require that CloudFront fetches objects from your origin using the protocol that the viewer used to request the objects. For example, when you use this CloudFront setting and when the viewer uses HTTPS to request an object from CloudFront, CloudFront also uses HTTPS to forward the request to your origin. When your origin is an Amazon S3 bucket, this is the default setting and cannot be changed.

If you're using an HTTP server as your origin, and if you want to use HTTPS both between viewers and CloudFront and between CloudFront and your origin, you must install an SSL certificate on the HTTP server that is signed by a third-party certificate authority, for example, VeriSign or DigiCert.

> **Caution**
>
> If the origin server returns an invalid certificate or a self-signed certificate, or if the origin server returns the certificate chain in the wrong order, CloudFront drops the TCP connection, returns HTTP error code 503, and sets the `X-Cache` header to `Error from cloudfront`.

You can, however, use a self-signed certificate if you only configure cache behaviors to require HTTPS between viewers and CloudFront, and you allow HTTP between CloudFront and the origin.

How CloudFront Works with HTTPS Connections

The following example of how CloudFront works with HTTPS connections assumes the following:

- Your CloudFront distribution has one cache behavior (the default cache behavior) and one origin.
- You have configured your distribution to use HTTPS between viewers and CloudFront and between CloudFront and your origin.
- Your origin has an SSL certificate that was signed by a third-party certificate authority.

The process works basically the same way whether your origin server is an Amazon S3 bucket or an HTTP server.

CloudFront Process for Serving Objects Using HTTPS

1. A viewer submits an HTTPS request to CloudFront. There's some SSL negotiation here between the viewer and CloudFront. In the end, the viewer submits the request in an encrypted format.
2. If the object is in the CloudFront edge cache, CloudFront encrypts the object and returns it to the viewer, and the viewer decrypts it.
3. If the object is not in the CloudFront cache, CloudFront performs the SSL negotiation with your origin and, when the negotiation is complete, forwards the request to your origin in an encrypted format.
4. Your origin decrypts the request, encrypts the requested object, and returns the object to CloudFront.
5. CloudFront decrypts the object, re-encrypts it, and forwards the object to the viewer. CloudFront also saves the object in the edge cache so the object is available next time it's requested.
6. The viewer decrypts the object.

How to Require HTTPS for Communication Between Viewers, CloudFront, and Your Origin

You can configure CloudFront to require HTTPS for communication between viewers and CloudFront and, optionally, between CloudFront and your origin.

Note
To ensure that objects are encrypted from the origin to CloudFront edge caches and from edge caches to viewers, use only HTTPS. If you ever configure CloudFront to get objects from your origin using HTTP, CloudFront adds the objects to the edge cache and continues to serve them to viewers until the objects expire, or until you remove or replace them. For more information about removing or replacing objects in a distribution, see Adding, Removing, or Replacing Objects in a Distribution (p. 64).

To Require HTTPS for Communication Between Viewers, CloudFront, and Your Origin

1. If you're using an HTTP server as your origin and if you don't already have an SSL certificate for the server, get and install an SSL certificate from a third-party certificate authority such as VeriSign or DigiCert.

   Note
   If you're using an Amazon S3 bucket, Amazon S3 provides an SSL certificate.

   For more information about getting and installing an SSL certificate, refer to the documentation for your HTTP server software and to the documentation for the third-party certificate authority.
2. To require that viewers use HTTPS when communicating with CloudFront, create or update one or more cache behaviors in your distribution to have the following settings:

- **CloudFront Console:** For **Viewer Protocol Policy**, specify **HTTPS Only**.
- **CloudFront API:** For `ViewerProtocolPolicy`, specify `https-only`.

For information about using the CloudFront console to update a download distribution, see Listing, Viewing, and Updating CloudFront Distributions (p. 55).

For information about using the CloudFront API to update a download distribution, go to PUT Distribution Config in the *Amazon CloudFront API Reference*.

3. *Optional:* To require that CloudFront uses HTTPS when communicating with your origin, create or update one or more origins in your distribution to have the following settings:

- **CloudFront Console:** For **Origin Protocol Policy**, specify **Match Viewer**.
- **CloudFront API:** For `OriginProtocolPolicy`, specify `match-viewer`.

When your origin is an Amazon S3 bucket, **Match Viewer** is the default setting and cannot be changed.

4. Confirm the following:

- The path pattern in each cache behavior applies only to the requests for which you want viewers to use HTTPS.
- The cache behaviors are listed in the desired order. For more information, see Path Pattern (p. 36).
- The cache behaviors are routing requests to the origins for which you have configured an **Origin Protocol Policy** of **Match Viewer**, if applicable.
- The origins have valid certificates signed by a third-party certificate authority.

Alternate Domain Names (CNAMEs) and HTTPS

If you want viewers to use HTTPS to access your objects, you must use the CloudFront domain name that is assigned to your distribution in your URLs. However, you can add one or more alternate domain names to a download distribution and then:

- Use HTTPS when the links to your objects use the CloudFront domain name, for example,
  `https://d111111abcdef8.cloudfront.net/image.jpg`.
- Use alternate domain names when the links to your objects use HTTP, for example,
  `http://www.example.com/image/image.jpg`.

If you want to use both alternate domain names and HTTPS in the same CloudFront download distribution (but not for the same objects), you must create at least one cache behavior for which the value of **Viewer Protocol Policy** is **HTTP and HTTPS** (`allow-all` if you're using the CloudFront API).

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.

Using IAM to Control Access to CloudFront Resources

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.

Download Distributions

API Actions for Download Distributions	Canonical Name
POST Distribution	CreateDistribution
GET Distribution	GetDistribution
GET Distribution Config	GetDistributionConfig
PUT Distribution Config	UpdateDistribution
GET Distribution List	ListDistributions
DELETE Distribution	DeleteDistribution

Streaming Distributions

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

Invalidations

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

Origin Access Identities

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.

```
{
    "Version": "2012-10-17",
    "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.

```
{
   "Version": "2012-10-17",
   "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.

```
{
   "Version": "2012-10-17",
   "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.

```
{
   "Version": "2012-10-17",
   "Statement":[{
      "Effect":"Allow",
      "Action":["cloudfront:*"],
      "Resource":"*",
      "Condition":{
         "Bool":{
            "aws:SecureTransport":"true"
            }
         }
      }
   ]
}
```

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

❶	Your end users use your application or website. In this graphic, you have two different websites, A and B, each using a different CloudFront distribution (Distribution A and Distribution B).
❷	Your end users send requests for content, and CloudFront routes each request to the appropriate edge location.
❸	CloudFront writes data about each request to a log file specific to that distribution. 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.
❹	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.

Each entry in a log file gives details about a single end user request for an object. For more information about log file format, see Log File Format (p. 146).

You can store log files for a distribution either in the same Amazon S3 bucket as your origin server or in a different bucket. (To simplify maintenance, we recommend that you use a separate bucket.) You can also store the log files for multiple distributions in the same bucket. When you enable logging for a particular distribution, you can specify an optional prefix for the names of your log files.

If no users access your content during a given hour, you don't receive any log files for that hour.

Analyzing Access Logs

Because logs for a single stream can get recorded in multiple files, we recommend that 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
We recommend that you 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.

For more information about CloudFront access logs, including recommendations for tools that you can use to analyze access logs, see Using CloudFront Logging (p. 252).

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 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 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 Change Logging Settings

You can enable or disable logging, change the Amazon S3 bucket where your logs are stored, and change the prefix for log files using the CloudFront console or using the CloudFront API:

* For information about updating a download or a streaming distribution using the CloudFront console, see Listing, Viewing, and Updating CloudFront Distributions (p. 55).
* For information about updating a download distribution using the CloudFront API, go to PUT Distribution Config in the *Amazon CloudFront API Reference*.
* For information about updating a streaming distribution using the CloudFront API, go to PUT Streaming Distribution Config in the *Amazon CloudFront API Reference*.

Your changes to logging settings take effect within 12 hours.

To use the CloudFront API to change access log settings for:

* download distributions, you must use the 2009-04-02 or later version of the API.
* streaming distributions, you must use the 2010-05-01 or later version of the API.
* cookies, you must use the 2012-07-01 or later version of the API.

> **Note**
> To enable easier listing of keys in a bucket, Amazon S3 users commonly use a prefix followed
> by a slash (/) as a delimiter.

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.*

File Name Format and Timing of File Delivery

The filename follows this format (with the date and hour in UTC):

```
{bucket-name}.s3.amazonaws.com/{optional-prefix}{distribution-ID}.{YYYY}-{MM}-{DD}-{HH}.{unique-ID}.gz
```

For example, if your bucket name is `mylogs` and your prefix is `myprefix/`, your filenames look similar
to this:

```
mylogs.s3.amazonaws.com/myprefix/EMLARXS9EXAMPLE.2012-07-01-20.RT4KCN4SGK9.gz
```

CloudFront compresses each log file using gzip and saves it in the Amazon S3 bucket that you specify.
Typically, CloudFront saves log files within 24 hours after receiving the corresponding requests. Depending
on how many users access your content, CloudFront can save several log files per hour.

> **Note**
> If no users access your content during a given hour, you don't receive any log files for that hour.

Log File Format

Topics
* Download Distribution File Format (p. 147)
* Streaming Distribution Log File Format (p. 149)

Each entry in a log file gives details about a single end user request for an object. The log files for download
and for streaming distributions are not identical, but both types of log files:

* 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.

- Substitute 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.

URL-Encoded Value	Character
%3C	<
%3E	>
%22	"
%23	#
%25	%
%7B	{
%7D	}
%7C	|
%5C	\
%5E	^
%7E	~
%5B	[
%5D	]
%60	`
%27	'
%20	space

Download Distribution File Format

The log file for a download distribution includes the following fields in the listed order.

Field	Description
date	The date (UTC) on which the event occurred, for example, 2009-03-10.
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.

Field	Description
sc-bytes	Server to client bytes, for example, 1045619.
c-ip	Client IP, for example, 192.0.2.183.
cs-method	HTTP access method.
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-uri-stem	URI stem (for example, /images/daily-ad.jpg).
sc-status	HTTP status code (for example, 200). For more information, see Supported HTTP Status Codes for Custom Origin Servers (p. 85).
cs(Referer)	The referrer.
cs(User-Agent)	The user agent.
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 hyphen (-) in the `cs-uri-query` field for that request. The encoding standard is RFC 1738. For more information, see Log File Format (p. 146).
cs(Cookie)	The cookie header in the request, including name-value pairs and the associated attributes. If you enable cookie logging, CloudFront logs the cookies in all requests regardless of which cookies you choose to forward to the origin: none, all, or a whitelist of cookie names. When a request doesn't include a cookie header, the log file contains a single hyphen (-) in the `cs(Cookie)` field for that request. For more information about cookies, see How CloudFront Forwards, Caches, and Logs Cookies (p. 62).
x-edge-result-type	The result type of a request. Result types include: • `Hit`: CloudFront served the object to the viewer from the edge cache. • `RefreshHit`: CloudFront found the object in the edge cache but it had expired, so CloudFront contacted the origin to verify that the cache has the latest version of the object. • `Miss`: The object wasn't in the edge cache, so CloudFront requested the object from the origin and served it to the viewer. • `LimitExceeded`: The request was denied because a CloudFront limit was exceeded. • `CapacityExceeded`: CloudFront returned a 503 error because the edge location didn't have enough capacity at the time of the request to serve the object. • `Error`: The request resulted in a client error (4xx) or server error (5xx).
x-edge-request-id	An encrypted string that uniquely identifies a request.

Note
Question marks (?) in URLs and query strings are not included in the log.

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 cs(Cookie) x-edge-result-
type x-edge-request-id
07/01/2012 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) - zip=98101 RefreshHit MRVMF7KydIvxMWf
JIglgwHQwZsbG2IhRJ07sn9AkKUFSHS9EXAMPLE==
07/01/2012 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 zip=50158 Hit
xGN7KWpVEmB9Dp7ctcVFQC4E-nrcOcEKS3QyAez--06dV7TEXAMPLE==
```

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. As a result, 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 hyphen (-).

The following table describes the fields that are present in each record in the streaming distribution log file, regardless of the type of event. The fields appear in the log in the order listed.

Field	Description
date	Date (UTC) on which the event occurred.
time	Time when the server received the request (UTC), for example, 01:42:39.
x-edge-location	The edge location where the playback event occurred. 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.

Field	Description
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 (-). These fields appear in the log after the field listed above, and they appear in the order listed.

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 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. 7)

General Usage Data

In addition to the optional CloudFront access logs, you can get other, more general information about your AWS service usage at no cost. (For more information about access logs, see Access Logs (p. 143).)

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 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.

Troubleshooting

If you cannot view your CloudFront distribution, the following topics describe some common solutions.

Topics

Did you sign up for both CloudFront and Amazon S3?

To use Amazon CloudFront with an Amazon S3 origin, you must sign up for both CloudFront and Amazon S3, separately. For more information about signing up for CloudFront and Amazon S3, see Getting Started with CloudFront (p. 9).

Are your Amazon S3 bucket and object permissions set correctly?

If you are using CloudFront with an Amazon S3 origin, the original versions of your content are stored in an Amazon S3 bucket. The easiest way to use CloudFront with Amazon S3 is to make all your objects publicly readable in Amazon S3. To do this, you must explicitly enable public read privileges for each object you upload to Amazon S3.

If your content is not publicly readable, you need to create a CloudFront origin access identity so CloudFront can access it. For more information about CloudFront origin access identities, see Using an Origin Access Identity to Restrict Access to Your Amazon S3 Content (p. 92).

Object properties and bucket properties are independent. You must explicitly grant privileges to each object in Amazon S3. Objects do not inherit properties from buckets and object properties must be set independently of the bucket.

Is your alternate domain name (CNAME) correctly configured?

If you already have an existing CNAME record for your domain name, 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, not your Amazon S3 bucket. You can confirm that the CNAME record in your DNS system points to your distribution's domain name. To do so, 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.example.com`, and the relevant part of the response. Under `ANSWER SECTION`, see the line that contains `CNAME`. The CNAME record for your domain name is set up correctly if the value on the right side of CNAME 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.

```
[prompt]> dig images.example.com

; <<> DiG 9.3.3rc2 <<> images.example.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.example.com.       IN    A
;; ANSWER SECTION:
images.example.com. 10800 IN  CNAME  d111111abcdef8.cloudfront.net.
...
...
```

For more information about CNAMEs, see Using Alternate Domain Names (CNAMEs) (p. 52).

Are you referencing the correct URL for your CloudFront distribution?

Make sure the URL you're referencing uses your CloudFront distribution domain name (or your CNAME), not your Amazon S3 bucket or custom origin.

Amazon CloudFront Developer Guide
If you are working with a streaming distribution, are your
URL and your playback client correctly configured?

If you are working with a streaming distribution, are your URL and your playback client correctly configured?

Streaming distributions require you to use an RTMP protocol instead of HTTP, and you must make a few minor configuration changes to your playback client. For information about creating streaming distributions, see Creating Streaming Distributions (p. 22).

Do you need help troubleshooting a custom origin?

If you need AWS to help you troubleshoot a custom origin, we will probably need to inspect the `X-Amz-Cf-Id` header entries from your requests. If you are not already logging these entries, you might want to consider it for the future. For more information, see Requirements and Recommendations for Using Amazon EC2 and Other Custom Origins (p. 42).

Load Testing CloudFront

Traditional load testing methods don't work well with CloudFront because CloudFront uses DNS to balance loads across geographically dispersed edge locations and within each edge location. When a client requests content from CloudFront, the client receives a DNS response that includes a set of IP addresses. If you test by sending requests from a single client IP to just one of the IP addresses that DNS returns, your requests may resolve to a single server in one CloudFront edge location. Depending on the volume of data requested, testing in this way may overload that CloudFront server and degrade performance.

CloudFront is designed to scale for viewers that have different client IP addresses and different DNS resolvers across multiple geographic regions. To perform load testing that accurately assesses CloudFront performance, we recommend that you do all of the following:

- Send client requests from multiple geographic regions.
- Configure your test so each client makes an independent DNS request; each client will then receive a different set of IP addresses from DNS.
- For each client that is making requests, spread your client requests across the set of IP addresses that are returned by DNS, which ensures that the load is distributed across multiple servers in a CloudFront edge location.

Making API Requests

Topics

This section describes how to make REST requests to the CloudFront 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; for example, 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, CloudFront can 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/2012-07-01/<distribution>
```

Related Topics

- REST Requests (p. 158)

- 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
- Python
- 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, 2012-07-01. 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:

```
/2012-07-01/distribution
```

For more information about creating a distribution using the CloudFront API, go to POST Distribution in the *Amazon CloudFront API Reference*).

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. For more information, see Authenticating REST Requests (p. 163).	Yes

Header Name	Description	Required
`Content-Length`	Length of the message (without the headers) according to RFC 2616. Condition: Required if the request body itself contains information (most toolkits add this header automatically).	Conditional
`Content-Type`	The content type of the resource. Example: `text/plain`. 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. Condition: Required unless you provide the `x-amz-date` header. For more information about the request time stamp, see Request Time Stamp (p. 159).	Conditional
`Host`	The host being requested. The value must be `cloudfront.amazonaws.com` 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. Condition: Required if you do not provide the `Date` header. For more information, see Request Time Stamp (p. 159).	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 /2012-07-01/distribution HTTP/1.1
Host: cloudfront.amazonaws.com
Authorization: [AWS authentication string]
Date: Thu, 17 May 2012 19:37:58 GMT
[Other required headers]

<?xml version="1.0" encoding="UTF-8"?>
<DistributionConfig xmlns="http://cloudfront.amazonaws.com/doc/2012-07-01/">
   <CallerReference>example.com2012-04-11-5:09pm</CallerReference>
   <Aliases>
      ...
   </Aliases>
   <DefaultRootObject>index.html</DefaultRootObject>
   <Origins>
      ...
   </Origins>
   <CacheBehaviors>
      ...
   </CacheBehaviors>
   <Comment>example comment</Comment>
   <Logging>
      ...
   </Logging>
   <PriceClass>PriceClass_All</PriceClass>
   <Enabled>true</Enabled>
</DistributionConfig>
```

Related Topics

- Authenticating REST Requests (p. 163)

- REST Responses (p. 161)

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/2012-07-01/distribution/EDFDVBD6EX
AMPLE
x-amz-request-id: request_id


<?xml version="1.0" encoding="UTF-8"?>
<Distribution xmlns="http://cloudfront.amazonaws.com/doc/2012-07-01/">
   <Id>EDFDVBD6EXAMPLE</Id>
   <Status>InProgress</Status>
   <LastModifiedTime>2012-05-19T19:37:58Z</LastModifiedTime>
   <DomainName>d111111abcdef8.cloudfront.net</DomainName>
   <DistributionConfig>
      <CallerReference>example.com2012-04-11-5:09pm</CallerReference>
      <Aliases>
         ...
      </Aliases>
      <DefaultRootObject>index.html</DefaultRootObject>
      <Origins>
         ...
      </Origins>
      <CacheBehaviors>
         ...
      </CacheBehaviors>
      <Comment>example comment</Comment>
      <Logging>
         ...
      </Logging>
      <PriceClass>PriceClass_All</PriceClass>
      <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/2012-07-01/">
   <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. 158)
- Authenticating REST Requests (p. 163)

Authenticating REST Requests

Topics

Every request you make to the CloudFront 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, see Where Do I Go from Here? (p. 252) and 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 S3 REST requests, then the information in this topic will be familiar to you. Here 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 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 task list describes 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. 164).

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. 164).

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. 164).

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. 165).

   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. 166).

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, 17 May 2012 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. 164)) 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 example shows a string, a fake Secret Access Key, and an example of a Base64-encoded signature. Note that the credentials have been obfuscated, so calculating using these values does not produce the Base64-encoded signature listed in the table.

- **String:** Thu, 17 May 2012 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 `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 /2012-07-01/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/2012-07-01/">
    <CallerReference>example.com2012-04-11-5:09pm</CallerReference>
    <Aliases>
        ...
    </Aliases>
    <DefaultRootObject>index.html</DefaultRootObject>
    <Origins>
        ...
    </Origins>
    <CacheBehaviors>
        ...
    </CacheBehaviors>
    <Comment>example comment</Comment>
    <Logging>
        ...
    </Logging>
    <PriceClass>PriceClass_All</PriceClass>
    <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/2012-07-01/">
    <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, for geoblocking, and for RTMP streaming:

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.

> **Note**
> This tutorial provides an overview of how to integrate CloudFront with Adobe Flash Media Server running on an Amazon EC2 instance. For more information about how to manage and secure your Amazon EC2 instance, refer to the Amazon EC2 documentation. For more information about Adobe Flash Media Server options not covered in this tutorial, see Adobe Documentation (p. 186).

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. 168)
2. Creating an Amazon EC2 Key Pair (p. 169)
3. Subscribing to Adobe Flash Media Server (p. 169)
4. Creating an AWS CloudFormation Stack for Live Streaming (p. 170)
5. Verifying that Adobe Flash Media Server Is Running (p. 174)
6. Setting Up Adobe Flash Media Live Encoder to Publish a Live Stream (p. 174)
7. Embedding Flash Media Playback for an Amazon CloudFront Live HTTP Stream in a Web Application (p. 178)
8. Deleting an AWS CloudFormation Stack for Live Streaming (p. 180)

For frequently asked questions, see Frequently Asked Questions (p. 180).

For links to additional Adobe and AWS documentation, see Additional Documentation (p. 186).

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. 169). 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 streaming, skip to Subscribing to Adobe Flash Media Server (p. 169). 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 **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.

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**.
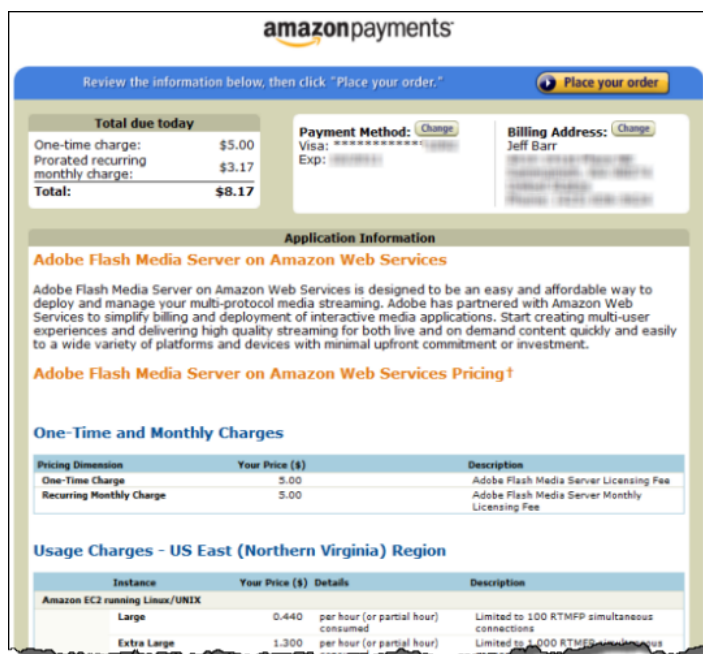
Creating an AWS CloudFormation Stack for Live Streaming

The following procedure uses an AWS CloudFormation template to create a stack that launches the AWS resources required by live streaming, including an Amazon EC2 instance.

> **Important**
> You incur hourly charges for an Amazon EC2 instance beginning when you create the AWS CloudFormation stack that deploys the Amazon EC2 instance. Charges continue to accrue until you delete the AWS CloudFormation stack regardless of whether you use the Amazon EC2 instance to stream live video. For more information, see Adobe Flash Media Server on Amazon Web Services Pricing. 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. For more information, see Deleting an AWS CloudFormation Stack for Live Streaming (p. 180).

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. 182).

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. 169), 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.

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. 170).
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. 168), 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. 186).

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**.
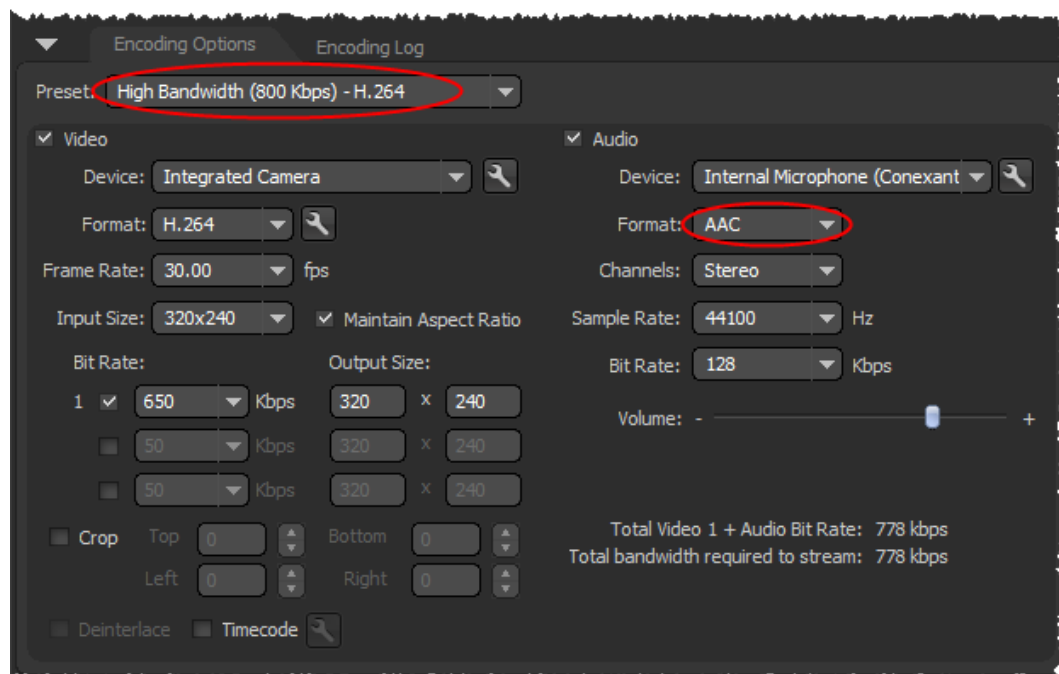
10. In the Video section of the Encoding Options tab, click the wrench icon on the right side of the Format list to open the Advanced Encoder Settings dialog box.
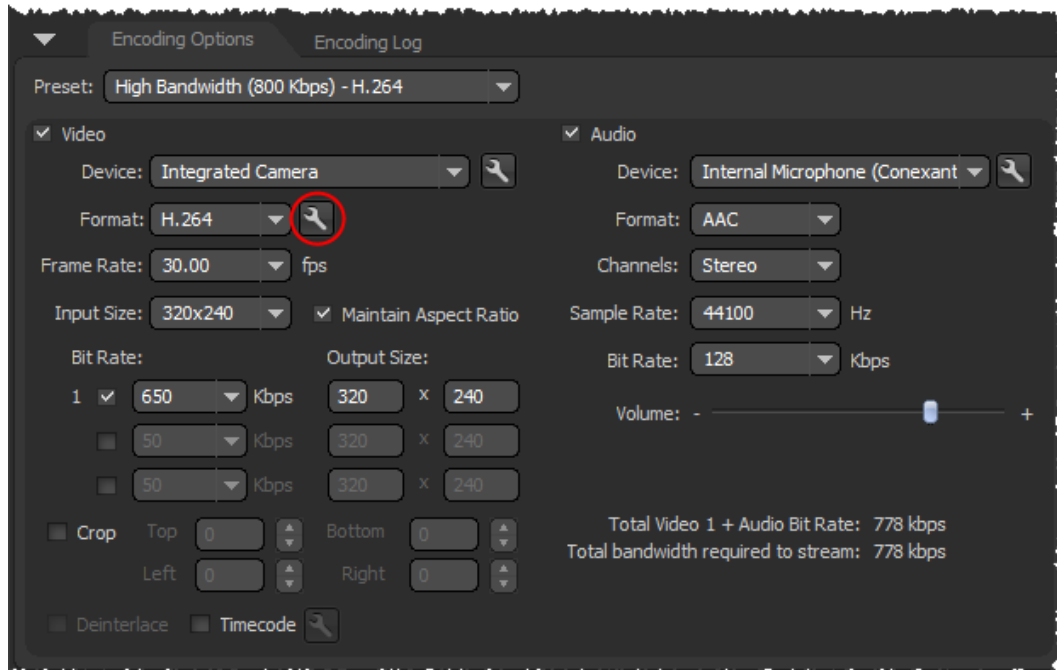
11. In the Advanced Encoder Settings dialog box, in the **Keyframe Frequency** list, click **4 Seconds**.

    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).
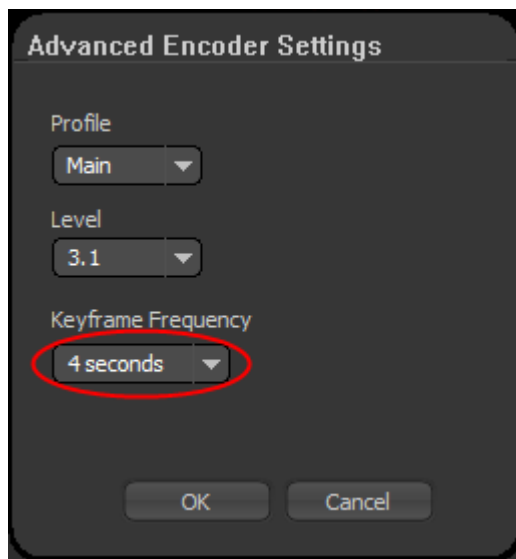
12. Click **OK** to save the setting and return to the main page. The value of the Preset list changes to `Custom`.
13. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
14. Check the checkbox for the stack that you created for live streaming.
15. In the bottom pane 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.
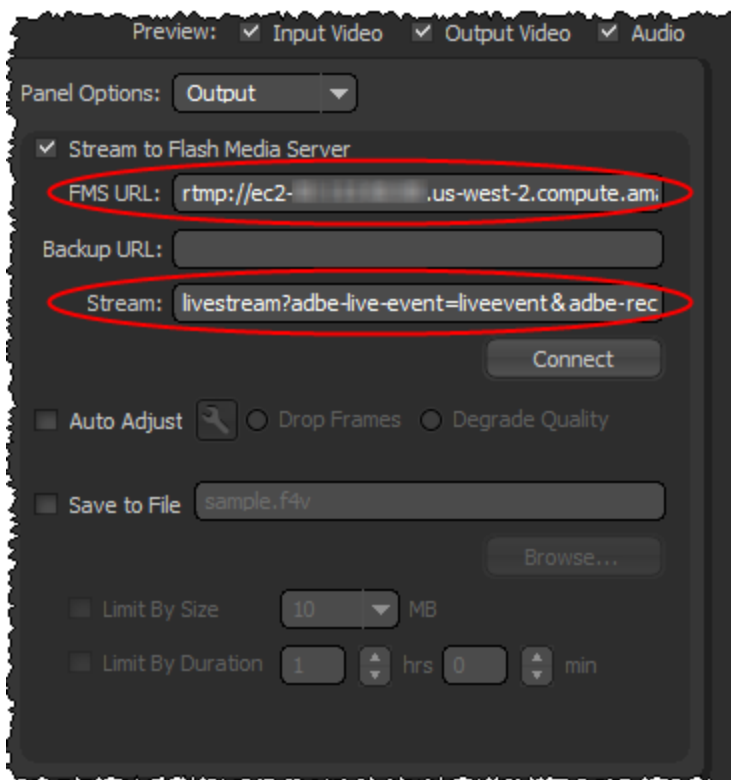
20. Uncheck the **Save to File** checkbox.

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. 178)

Amazon CloudFront Developer Guide
Embedding Flash Media Playback for an Amazon
CloudFront Live HTTP Stream in a Web Application

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

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.
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.

Amazon CloudFront Developer Guide
Embedding Flash Media Playback for an Amazon
CloudFront Live HTTP Stream in a Web Application

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.

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. 182).

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. 180)

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. 181)
- 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. 182)
- What is the price for live HTTP streaming using CloudFront and Adobe Flash Media Server 4.5? (p. 183)
- How can I create a CNAME alias for my Amazon EC2 instance or for my CloudFront distribution? (p. 183)
- How can I connect to the Flash Media Administration Console? (p. 183)
- Can I stream my live event both to Apple devices and to Flash Player–compatible devices? (p. 184)
- Does Flash Media Server 4.5 support HTML5? (p. 185)

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:
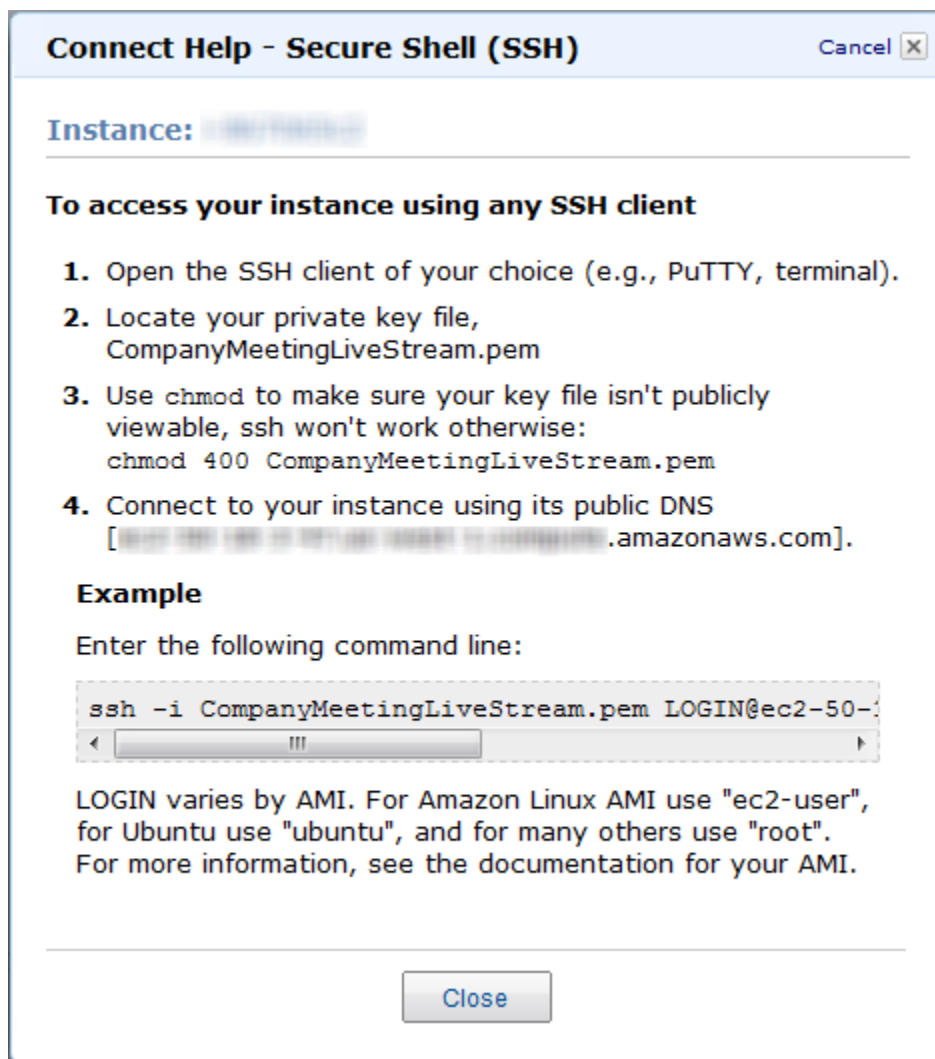
How do I update crossdomain.xml so my users can view the live stream using a Flash-based player that is hosted on my own domain?

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 Alternate Domain Names (CNAMEs) (p. 52).

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. 170).
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. 183). 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. 174) 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 **Region** list, 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. 174), 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 Flash Platform, http://www.adobe.com/devnet/adobe-media-server/articles/encoding_recommendations.html
- *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.

Note

Microsoft Expression Encoder 4 Pro with Service Pack 2 is not included in the Amazon EC2 Amazon Machine Image (AMI) that is running Windows IIS Media Services, and it is not a free download. For 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.

Note

This tutorial provides an overview of how to integrate CloudFront with Microsoft Live Smooth
Streaming running on an Amazon EC2 instance. For more information about how to manage
and secure your Amazon EC2 instance, refer to the Amazon EC2 documentation. For more
information about Microsoft Live Smooth Streaming options not covered in this tutorial, see
Microsoft Documentation (p. 204).

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. 189)
2. Creating an Amazon EC2 Key Pair (p. 189)
3. Creating an AWS CloudFormation Stack for Live Smooth Streaming (p. 190)
4. Verifying that Your Amazon EC2 Windows Server Instance Is Running (p. 193)
5. Getting Your Windows Password (p. 193)
6. Encoding Your Live Stream (p. 196)
7. Viewing Your Live Smooth Stream (p. 201)
8. Deleting Your AWS CloudFormation Live Smooth Streaming Stack (p. 201)

For frequently asked questions, see Frequently Asked Questions (p. 202).

For links to additional Microsoft and AWS documentation, see Additional Documentation (p. 204).

Creating an Amazon Web Services Account

If you already have an AWS account, skip to Creating an Amazon EC2 Key Pair (p. 189). 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. 189)

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. 190).
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 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, including an Amazon EC2 instance.

> **Important**
> You incur hourly charges for an Amazon EC2 instance beginning when you create the AWS
> CloudFormation stack that deploys the Amazon EC2 instance. Charges continue to accrue until
> you delete the AWS CloudFormation stack regardless of whether you use the Amazon EC2
> instance to stream live video. For more information, see Pricing on the Amazon Elastic Compute
> Cloud (Amazon EC2) detail page. 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-streaming event, and stops the AWS charges for the resources. For more information, see
> Deleting Your AWS CloudFormation Live Smooth Streaming Stack (p. 201).

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. 189), 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. 190).
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. 189).
2. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
3. 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**.
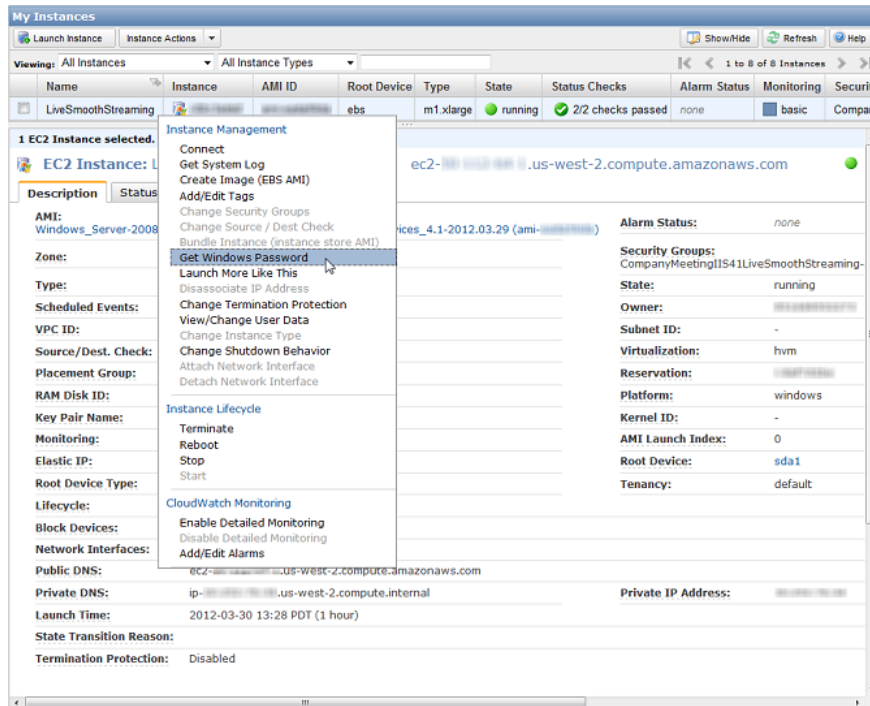
6. On the Retrieve Default Windows Administrator Password page, click **Browse**, and browse to the location on your computer where you saved the .pem file.
7. Select the .pem file, and the contents of the file appear in the window.

8. Click **Decrypt Password**.

9. Write down the password. You'll need it to connect to the Amazon EC2 instance.
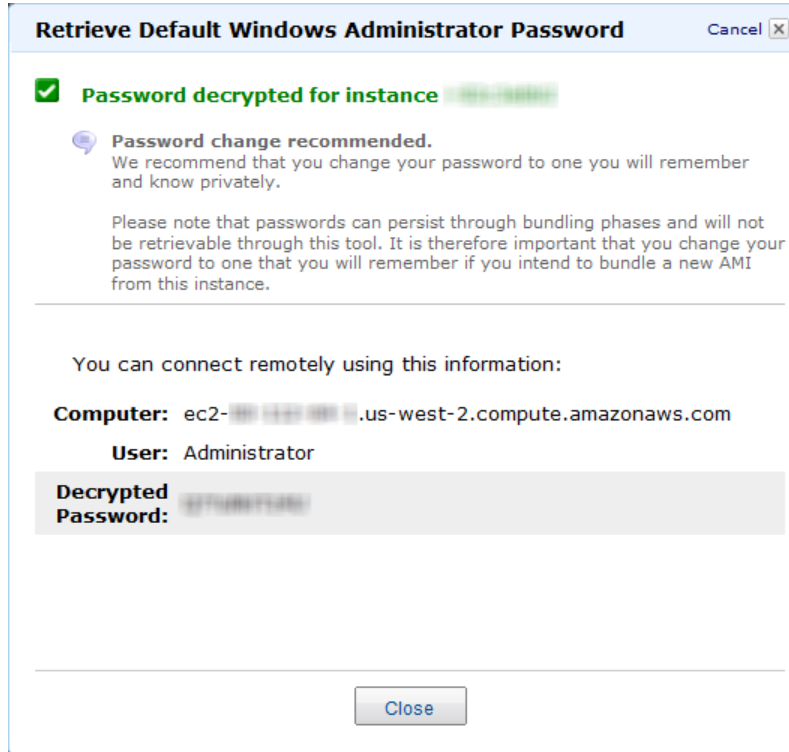
10. *Optional but recommended:* Log into the Windows Server instance that you just launched, and change the password for the default Windows Server account. The username is Administrator.

   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. 203). 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. 203).

Next: Encoding Your Live Stream (p. 196)

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**.
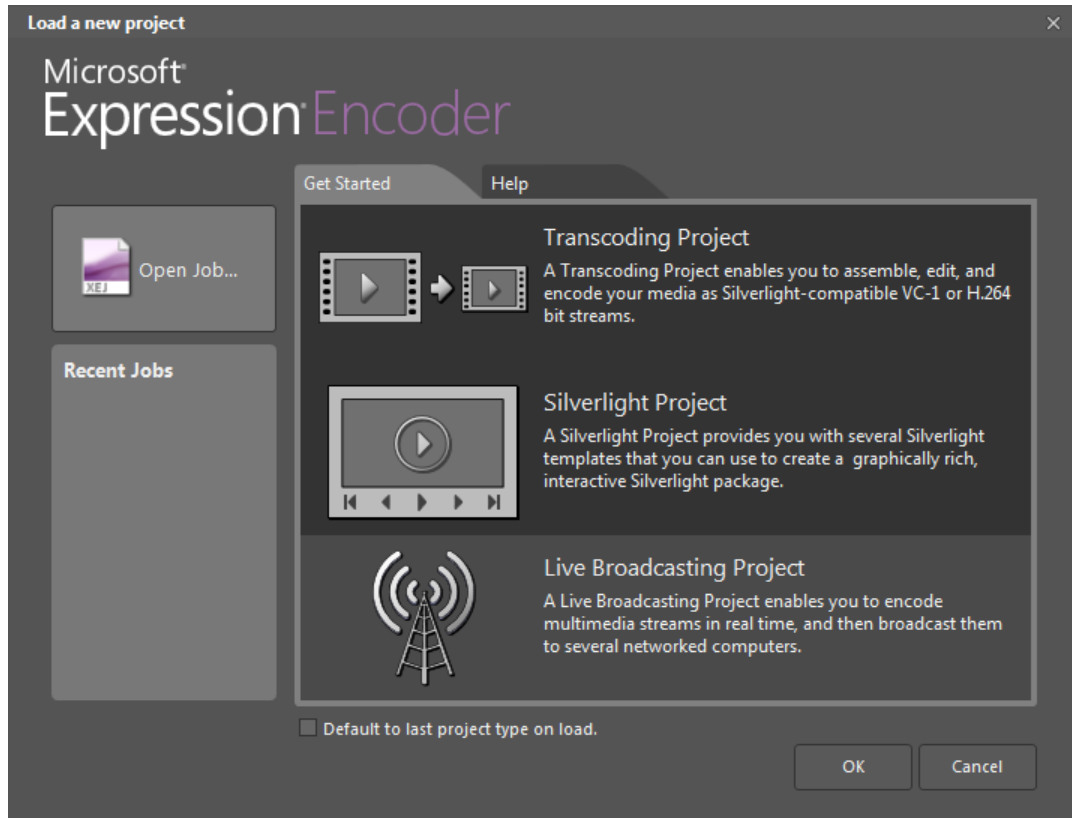
4. Click **Add a Live Source** to use for your live broadcast.
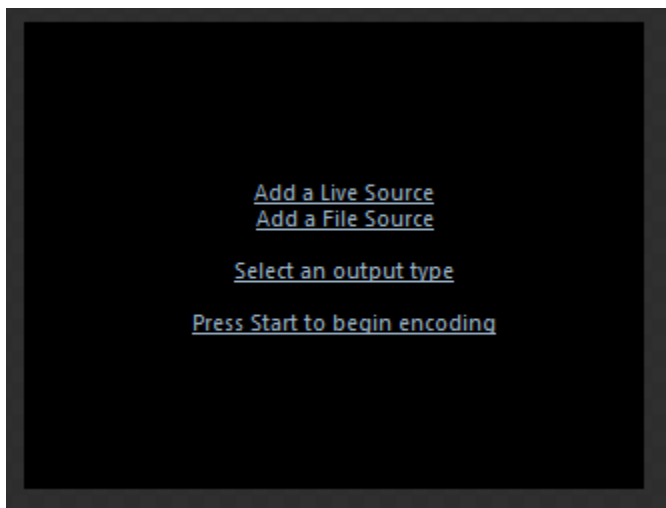
Note
You can connect multiple camera devices, such as USB webcams or FireWire (IEEE 1394) digital video cameras. Although you can connect multiple live sources, you can stream only one at a time. For more information about setting up sources for live broadcasts, go to Set Live Sources on the Microsoft 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.
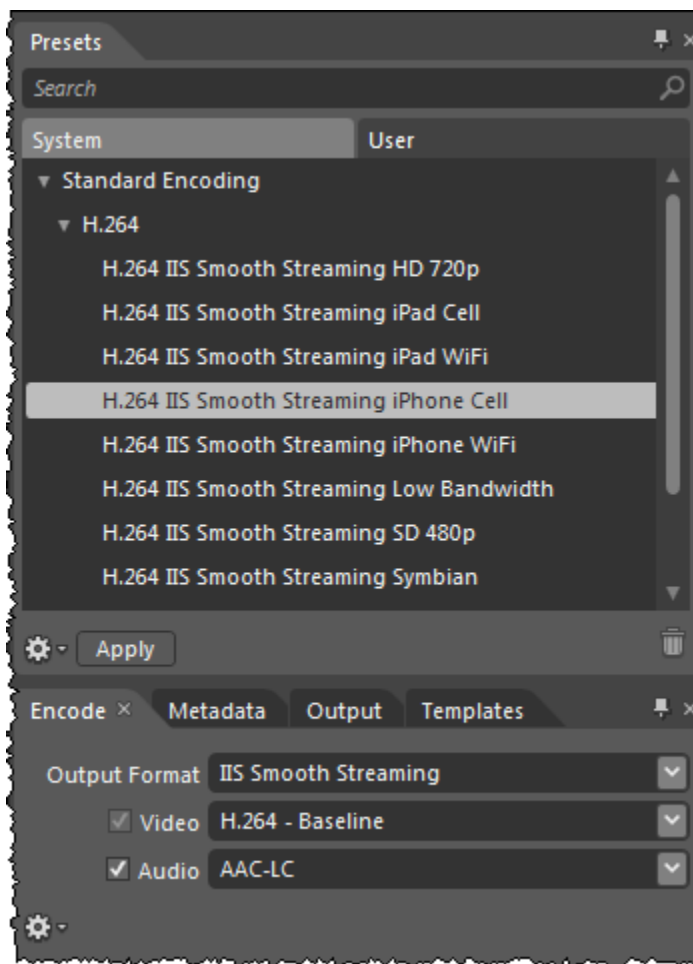
When you click **Apply**, the **Output Format**, **Video**, and **Audio** settings on the **Encode** tab are automatically updated with the values in the encoding preset that you selected.

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.
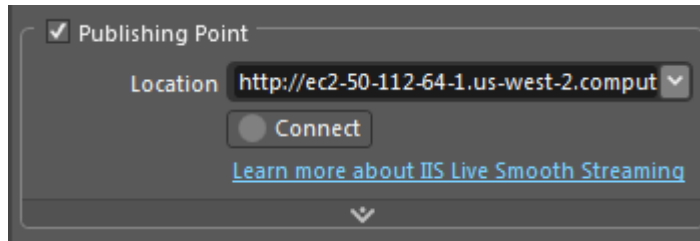
12. Click **Connect** to initiate a connection to the publishing point on your Windows server.

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. 193).

    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.
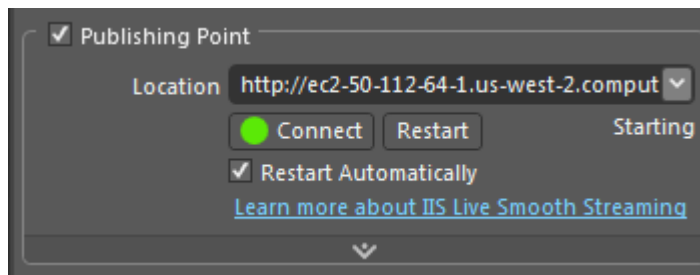
    > **Note**
    > The Starting state means that the publishing point is ready to receive live streams. When the live source connects to the publishing point and begins pushing content to it, the Starting state changes to Started, meaning that the publishing point is receiving the live streams.

    > **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.
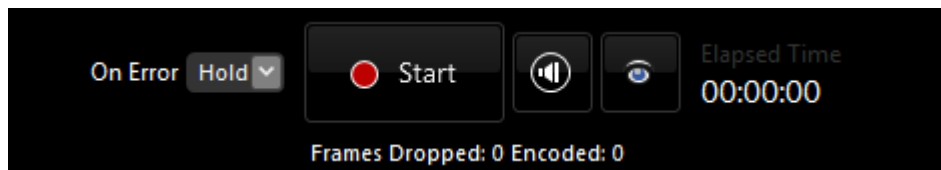
As the broadcast runs, you can monitor statistics and connections data in the corresponding panels. For more information about how to monitor this data, see the following topics on the Microsoft Expression website:

- 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?

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 Alternate Domain Names
(CNAMEs) (p. 52).

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 **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. 190).
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
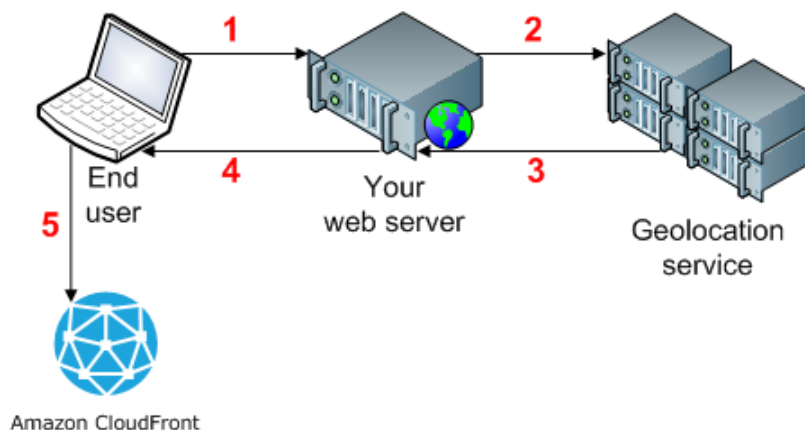
Using CloudFront and a third-party geolocation service to restrict access to your content from your application layer gives you full control over your end user's experience. For end users whose access is blocked, your application can display a meaningful message instead of returning an error code. You 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. 207)

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 Serving Private Content through CloudFront (p. 87).

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. 207)
   - Sample Code for MaxMind (p. 217)

   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. 207)

- .NET Sample Code for Digital Element (p. 211)

- PHP Sample Code for Digital Element (p. 215)

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 {

      String ip = null;
      StringTokenizer st = null;
      PrintWriter out = response.getWriter();
```

```java
        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.

```csharp
<%@ 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
    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.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Cre
ateSignatureInCSharp.html
```

```
    /// </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.

```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 = '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);
$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);

   // 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='
```

```php
       . $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.

```
/*
 * 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.
 */
```

```java
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;
}
```

```php
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.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Cre
ateSignatureInCSharp.html
    /// </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/

On-Demand Video Streaming Using CloudFront and Adobe Flash Player

When you stream media files using CloudFront, you provide both your media file and the media player with which you want end users to play the media file. To use Adobe Flash Player to stream media files with CloudFront, perform the procedures in the following topics:

This tutorial uses Adobe Flash Builder version 4.6 to generate the files necessary to stream a video using Adobe Flash Player. For more information about Flash Builder, go to the Flash Builder page on the Adobe website. For information about downloading a free trial version of Adobe Flash Builder, go to the Downloads/Adobe Flash Builder 4.6 page.

For a list of the codecs that Flash Player supports, go to the Supported codecs | Flash Player page on the Adobe website.

For more information about streaming media using CloudFront, see Working with Streaming Distributions (p. 43).

Creating an Amazon S3 Bucket

You can upload your media files and your media player files to the same Amazon S3 bucket or to separate buckets. For this tutorial, you'll create one bucket for both a media file and and the Flash Player media player files. You'll upload the files later in the process, after you create the Adobe Flash Player files.

To create an Amazon S3 bucket

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, click **Create Bucket**.
3. In the **Create Bucket** dialog, enter a bucket name.

    Important
    For your bucket to work with CloudFront, the name must conform to DNS naming requirements. For more information, go to Bucket Restrictions and Limitations in the *Amazon Simple Storage Service Developer Guide*.

4. Select a region for your bucket. By default, Amazon S3 creates buckets in the US Standard region. We recommend that you choose a region close to you to optimize latency, minimize costs, or to address regulatory requirements.
5. Click **Create**.

Creating CloudFront Download and Streaming Distributions

To configure CloudFront to stream a media file, you need a CloudFront streaming distribution. For this tutorial, you'll also create a CloudFront download distribution to access the .hlml file that Adobe Flash Builder creates. Perform the following two procedures.

To create a CloudFront download distribution

1. Open the Amazon CloudFront console at https://console.aws.amazon.com/cloudfront/.
2. Click **Create Distribution**.
3. On the first page of the **Create Distribution Wizard**, accept the default selection, **Download**, and click **Continue**.

4. On the second page of the wizard, click in the **Origin Domain Name** field, and select the Amazon S3 bucket that you created in the procedure To create an Amazon S3 bucket (p. 230). If you have a lot of Amazon S3 buckets, you can type the first few characters of the bucket name to filter the list.

5. Accept the default values for the remaining fields, and click **Create Distribution**.

6. After CloudFront creates your distribution, the value of the **Status** column for your distribution will change from **InProgress** to **Deployed**. This should take less than 15 minutes.

   The domain name that CloudFront assigns to your distribution appears in the list of distributions. (It also appears on the **General** tab for a selected distribution.)

To create a CloudFront streaming distribution

1. In the CloudFront console, click **Create Distribution**.

2. In the **Create Distribution Wizard**, click **Streaming**, and click **Continue**.

3. On the second page of the wizard, click in the **Origin Domain Name** field, and select the Amazon S3 bucket that you created in the procedure To create an Amazon S3 bucket (p. 230). If you have a lot of Amazon S3 buckets, you can type the first few characters of the bucket name to filter the list.

4. Accept the default values for the remaining fields on the **Create Distribution** page, and click **Create Distribution**.

5. After CloudFront creates your distribution, the value of the **Status** column for your distribution will change from **InProgress** to **Deployed**. This should take less than 15 minutes.

   The domain name that CloudFront assigns to your distribution appears in the list of distributions. The domain name also appears on the **General** tab for a selected distribution.

Creating a Flash Project Using Adobe Flash Builder

You can use Adobe Flash Builder to automatically create a Flash project, which contains all of the files necessary to play a media file using Adobe Flash.

To create a Flash project using Adobe Flash Builder

1. Start Adobe Flash Builder.
2. On the Flash Builder **File** menu, click **New > Flex Project**.
3. Enter the following values:

- **Project name:** Enter a name for your project, for example, **CloudFrontStreaming**.
- **Folder:** Specify where you want Flash Builder to save the files for this project. If you don't want to use the default location, uncheck the **Use default location** checkbox, and choose another location.

  Make note of the location; you'll need it later in the process.
- **Application type:** Accept the default value, **Web**.
- **Flex SDK version:** Accept the default value, **Use default SDK**.

4. To create the project, click **Finish**.

   After Flash Builder creates the project, a new tab that has the name of your project appears in the Flash Builder user interface. The **Source** button on the **<project-name>** tab is selected, and the **Source** page contains several lines of XML code.

5. Delete the default XML code on the **Source** page.

6. Copy the following XML code, and paste it into the blank **Source** page in Adobe Flash Builder.

```xml
<?xml version="1.0" encoding="utf-8"?>
    <s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
        xmlns:s="library://ns.adobe.com/flex/spark"
        xmlns:mx="library://ns.adobe.com/flex/mx" minWidth="955" min
Height="600">
    <fx:Declarations>
        <!-- Place non-visual elements here, for example, services and value
 objects -->
    </fx:Declarations>
    <fx:Script>
        <![CDATA[
            import mx.events.FlexEvent;
          import org.osmf.net.StreamingURLResource; import org.osmf.net.FMSURL;

            protected function vp_preinitializeHandler(event:FlexEvent): void
            {
            var myURL:StreamingURLResource = new StreamingURLResource("rt
mp://STREAMING-DISTRIBUTION-DOMAIN-NAME/cfx/st/mp4:VIDEO-FILE-NAME-WITHOUT-
EXTENSION");
                myURL.urlIncludesFMSApplicationInstance = true;
                myVideoPlayer.source = myURL;
            }
        ]]>
    </fx:Script>
    <s:VideoPlayer id="myVideoPlayer" autoPlay="true" preinitialize="vp_prein
itializeHandler(event)"  x="32" y="52"/>
</s:Application>
```

7. In the XML code that you pasted into the **Source** page, replace the following values:

- Replace STREAMING-DISTRIBUTION-DOMAIN-NAME with the CloudFront domain name for your streaming distribution, for example, s5c39gqb8ow64r.cloudfront.net.
- Replace VIDEO-FILE-NAME-WITHOUT-EXTENSION with the name of your video file, but exclude the filename extension. For example, if the name of your video is my-vacation.mp4, enter only my-vacation.

8. Save your changes.

9. On the Flash Builder **Project** menu, click **Export Release Build**.

10. In the **Export Release Build** dialog box, accept all default values, and click **Finish**.

Flash Builder creates the files for your project and saves them in the location that you specified in Step 3.

Uploading Media and Flash Builder Files to an Amazon S3 Bucket

When you use Adobe Flash Builder to generate the files for streaming media files, you upload media files and Flash Builder files to the same Amazon S3 bucket.

To upload your media and Flash Builder files to an Amazon S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** pane, select your bucket, and click **Upload**.
3. On the **Upload - Select Files** page, click **Add Files**, and add the following files:

   • Your media file
   • The files that Flash Builder generated when you performed the procedure To create a Flash project using Adobe Flash Builder (p. 233). Upload only the files in the `bin-release` directory. You can exclude the files in the `bin-release/history` subdirectory.
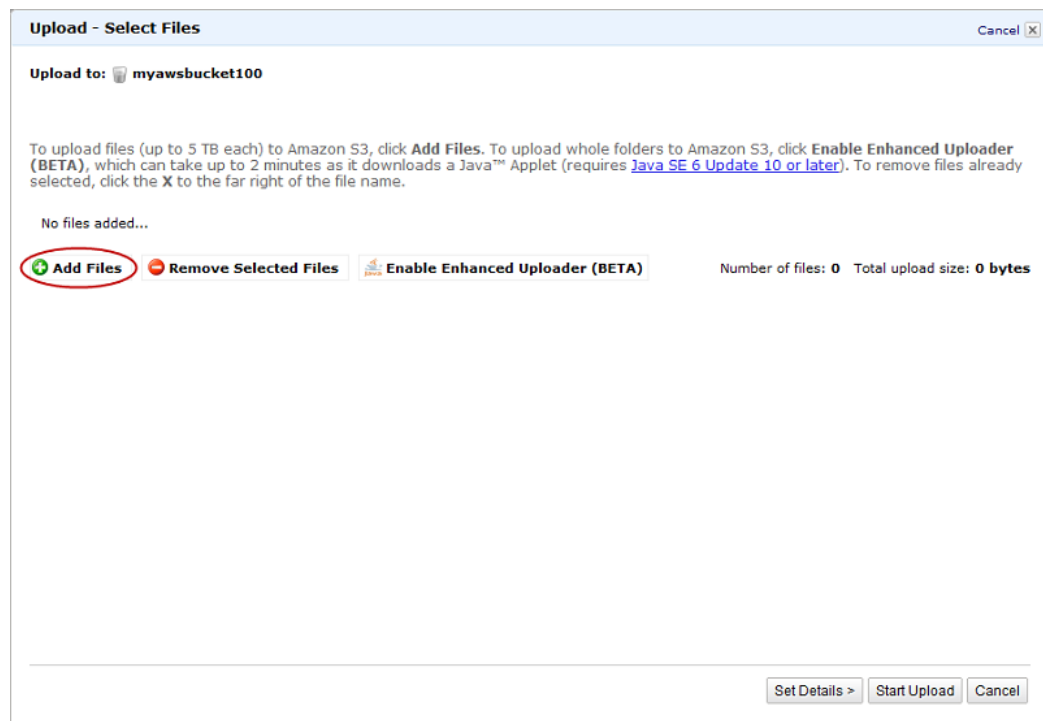
4. Grant public read permissions for the files that you added in the previous step.

   a. Click **Set Details**.
   b. On the **Set Details** page, click **Set Permissions**.
   c. On the **Set Permissions** page, click **Make everything public**.

5.   Click **Start Upload**.

Playing the Media File

To play the media file, you display the HTML file that Flash Builder created for your project and that you uploaded to your Amazon S3 bucket.

To play the media file

1.   Enter the CloudFront URL of the HTML file that Flash Builder created for your project by concatenating the following values:

     ```
     http://domain-name-for-your-CloudFront-distribution/HTML-file-name
     ```

     For example:

     ```
     http://d111111abcdef8.cloudfront.net/CloudFrontStreaming.html
     ```

2.   In the video player, click the arrow button.

     The video should begin to play.

On-Demand Video Streaming Using CloudFront and Flowplayer for Adobe Flash

When you stream media files using CloudFront, you provide both your media file and the media player with which you want end users to play the media file. To use the Flowplayer for Adobe Flash media player to stream media files with CloudFront, perform the procedures in the following topics:

> **Note**
> To stream video using CloudFront and Flowplayer for Adobe Flash, your users must enable Javascript in their browsers.

This tutorial is based on version 3.2.12 of Flowplayer for Adobe Flash. For more information about Flowplayer Flash, go to the Flowplayer Flash website. For a list of the video formats that Flowplayer Flash supports, go to Video formats in the Flowplayer developer documentation about the Flowplayer development environment.

> **Note**
> Flowplayer has released an HTML 5 version of their media player. The following procedures only work with Flowplayer Flash, not with Flowplayer HTML5.

For more information about streaming media using CloudFront, see Working with Streaming Distributions (p. 43).

Uploading Media and Flowplayer Files to an Amazon S3 Bucket

You can upload your media files and your media player files to the same Amazon S3 bucket or to separate buckets. For this tutorial, you'll upload an .mp4 media file and the Flowplayer media player files to the same bucket.

To upload media and Flowplayer files to an Amazon S3 bucket

1. Download the following files from the Flowplayer website:

   • The Flowplayer media player. After you download Flowplayer, extract the contents of the .zip file.
   • flowplayer.rtmp-3.2.10.swf. This is a plugin that allows Flowplayer to stream video using the RTMP protocol. The file is available on the RTMP page on the Flowplayer website.

2. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

3. In the Amazon S3 console, click **Create Bucket**.

4. In the **Create Bucket** dialog, enter a bucket name.

   > **Important**
   > For your bucket to work with CloudFront, the name must conform to DNS naming requirements. For more information, go to Bucket Restrictions and Limitations in the *Amazon Simple Storage Service Developer Guide*.

5. Select a region for your bucket. By default, Amazon S3 creates buckets in the US Standard region. We recommend that you choose a region close to you to optimize latency, minimize costs, or to address regulatory requirements.

6. Click **Create**.

7. Select your bucket in the **Buckets** pane, and click **Upload**.

8. On the **Upload - Select Files** page, click **Add Files**, and add the following files (the Flowplayer version numbers in your files may be different):

    *   flowplayer.controls-3.2.12.swf
    *   flowplayer-3.2.11.min.js
    *   flowplayer-3.2.12.swf
    *   flowplayer.rtmp-3.2.10.swf
    *   Your media file in .mp4 format
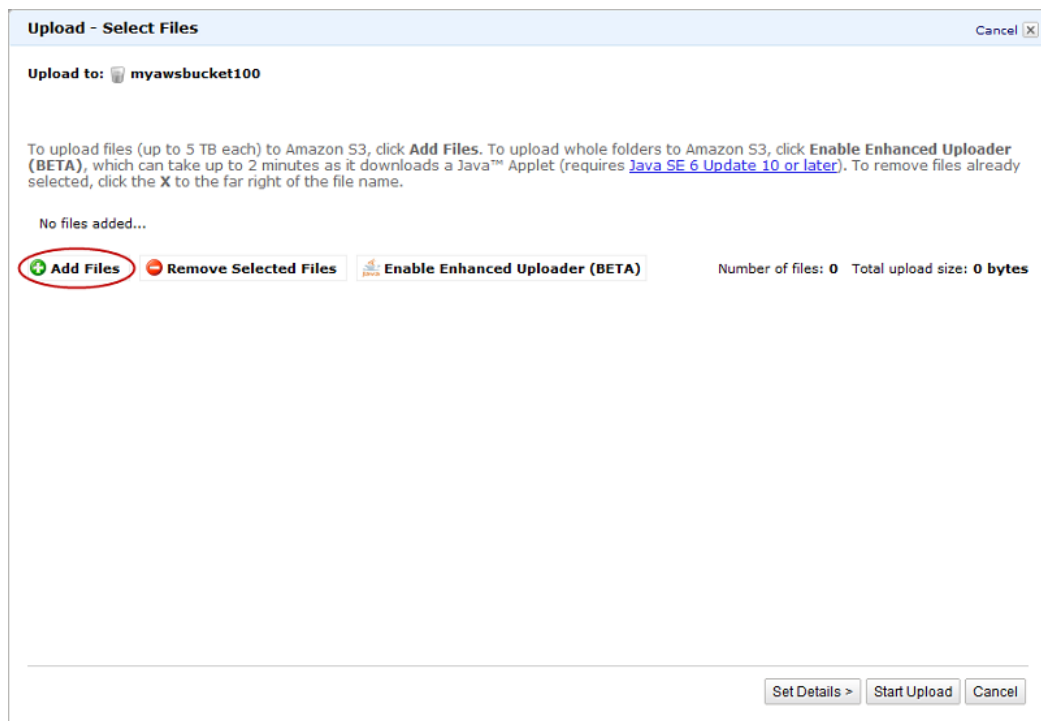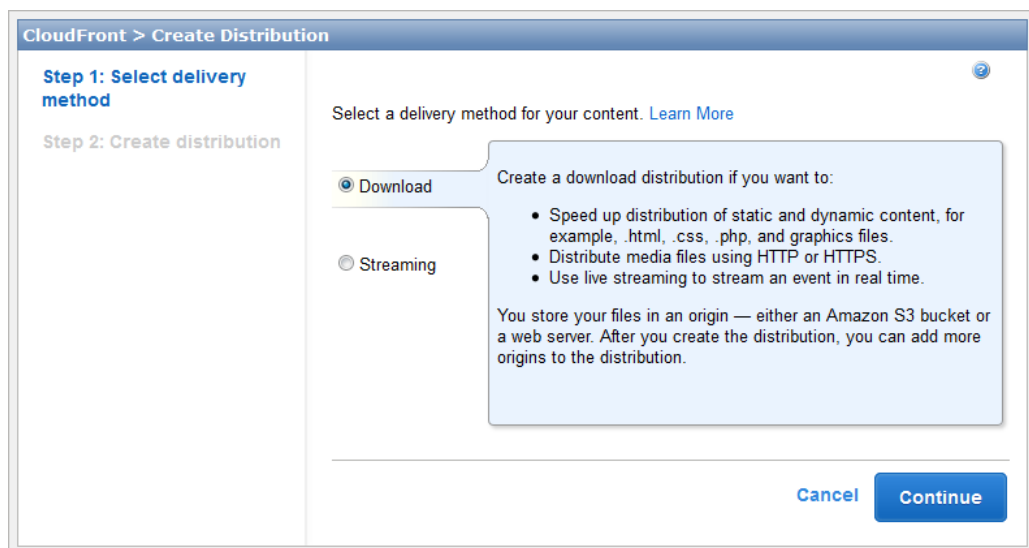
9. Grant public read permissions for the files that you added in the previous step.

    a. Click **Set Details**.
    b. On the **Set Details** page, click **Set Permissions**.
    c. On the **Set Permissions** page, click **Make everything public**.

10. Click **Start Upload**.

Creating CloudFront Download and Streaming Distributions

To configure CloudFront to stream a media file, you need a CloudFront download distribution for the Flowplayer files and a streaming distribution for the media file. Perform the following two procedures to create a download distribution and a streaming distribution.

To create a CloudFront download distribution for your Flowplayer files

1. Open the Amazon CloudFront console at https://console.aws.amazon.com/cloudfront/.
2. Click **Create Distribution**.
3. On the first page of the **Create Distribution Wizard**, accept the default selection, **Download**, and click **Continue**.

4. On the second page of the wizard, click in the **Origin Domain Name** field, and select the Amazon S3 bucket that you created in the procedure To upload media and Flowplayer files to an Amazon S3 bucket (p. 237). If you have a lot of Amazon S3 buckets, you can type the first few characters of the bucket name to filter the list.

5. Accept the default values for the remaining fields, and click **Create Distribution**.

6. After CloudFront creates your distribution, the value of the **Status** column for your distribution will change from **InProgress** to **Deployed**. This should take less than 15 minutes.

   The domain name that CloudFront assigns to your distribution appears in the list of distributions. (It also appears on the **General** tab for a selected distribution.)

To create a CloudFront streaming distribution for your media file

1. In the CloudFront console, click **Create Distribution**.

2. In the **Create Distribution Wizard**, click **Streaming**, and click **Continue**.

3. On the second page of the wizard, click in the **Origin Domain Name** field, and select the Amazon S3 bucket that you created in the procedure To upload media and Flowplayer files to an Amazon S3 bucket (p. 237). If you have a lot of Amazon S3 buckets, you can type the first few characters of the bucket name to filter the list.

4. Accept the default values for the remaining fields on the **Create Distribution** page, and click **Create Distribution**.

5. After CloudFront creates your distribution, the value of the **Status** column for your distribution will change from **InProgress** to **Deployed**. This should take less than 15 minutes.

   The domain name that CloudFront assigns to your distribution appears in the list of distributions. The domain name also appears on the **General** tab for a selected distribution.

Embedding Video in an HTML Page

The following sample HTML file shows you how to stream a video using the download and streaming distributions that you created in . To use this sample to stream your video, perform the following steps:

1. Copy the HTML code below, and paste it into a text editor.

2. Review the comments in the HTML file, and replace the following placeholders with the applicable values:

   - DOWNLOAD-DISTRIBUTION-DOMAIN-NAME
   - VIDEO-FILE-NAME

• STREAMING-DISTRIBUTION-DOMAIN-NAME

3. Save the file with a .html filename extension, for example, `flowplayer-example.html`.
4. Open the .html file in a web browser, and play your video.

```
<HTML>
<HEAD>
<TITLE>Amazon CloudFront Streaming with Flowplayer</TITLE>
</HEAD>

<BODY>

<H1>This video is streamed by CloudFront and played in Flowplayer.</H1>

<!-- This HTML file plays an MP4 media file using Flowplayer.

Replace all instances of DOWNLOAD-DISTRIBUTION-DOMAIN-NAME with the
domain name of your CloudFront download distribution, for example,
d111111abcdef8.cloudfront.net (begins with "d").

Update the version number that appears in the flowplayer-version filenames
with the version number of the files that you downloaded from the Flowplayer
website.
The files may not have the same version number.

Ensure that URLs don't contain any spaces.
-->

<!-- Call the Flowplayer JavaScript file. -->
<script src="http://DOWNLOAD-DISTRIBUTION-DOMAIN-NAME/flowplayer-
3.2.11.min.js"></script>

<!-- Style section. Specify the attributes of the player
such as height, width, color, and so on.
-->
<style>
a.rtmp {
   display:block;
   width:720px;
   height:480px;
   margin:25px 0;
   text-align:center;
   background-color:black;
}
</style>

<!--  Replace VIDEO-FILE-NAME with the name of your .mp4 video file,
excluding the .mp4 filename extension. For example, if you uploaded a file
called my-vacation-video.mp4, enter my-vacation-video.

If you're streaming an .flv file, use the following format:
<a class="rtmp" href="VIDEO-FILE-NAME"/>
-->
<a class="rtmp" href="mp4:VIDEO-FILE-NAME"/>

<script type="text/javascript">
```

```
$f("a.rtmp", "http://DOWNLOAD-DISTRIBUTION-DOMAIN-NAME/flowplayer-3.2.12.swf",
 {
    // Configure Flowplayer to use the RTMP plugin for streaming.
    clip: {
       provider: 'rtmp'
    },

    // Specify the location of the RTMP plugin.
    plugins: {
       rtmp: {
          url: 'http://DOWNLOAD-DISTRIBUTION-DOMAIN-NAME/flowplayer.rtmp-
3.2.10.swf',

          // Replace STREAMING-DISTRIBUTION-DOMAIN-NAME with the domain name of
 your
          // CloudFront streaming distribution, for example,
s5c39gqb8ow64r.cloudfront.net.
          netConnectionUrl: 'rtmp://STREAMING-DISTRIBUTION-DOMAIN-NAME/cfx/st'

       }
    }
});
</script>

</BODY>
</HTML>
```

On-Demand Video Streaming Using CloudFront and JW Player

When you stream media files using CloudFront, you provide both your media file and the media player with which you want end users to play the media file. To use the JW Player media player from Longtail Video to stream media files with CloudFront, perform the procedures in the following topics:

This tutorial is based on the free edition of JW Player version 6.1. For more information about JW Player, go to the Longtail Video website. For a list of the video formats that JW Player supports, go to the JW Player Tech Specs page.

For more information about streaming media using CloudFront, see Working with Streaming Distributions (p. 43).

Uploading Media and JW Player Files to an Amazon S3 Bucket

You can upload your media files and your media player files to the same Amazon S3 bucket or to separate buckets. For this tutorial, you'll upload an .mp4 or .flv media file and the JW Player media player files to the same bucket.

To upload media and JW Player files to an Amazon S3 bucket

1. If you don't already have the files for the JW Player media player, download the player from the JW Player page on the Longtail Video website. Then extract the contents of the .zip file.
2. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
3. In the Amazon S3 console, click **Create Bucket**.
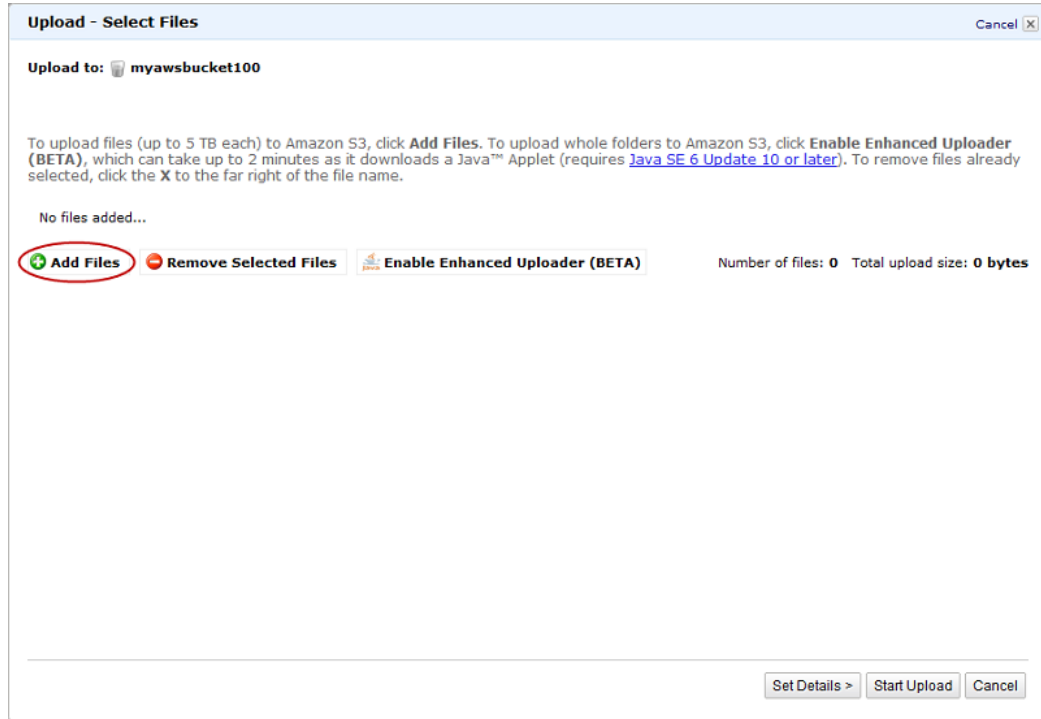4. In the **Create Bucket** dialog, enter a bucket name.

   > **Important**
   > For your bucket to work with CloudFront, the name must conform to DNS naming requirements. For more information, go to Bucket Restrictions and Limitations in the *Amazon Simple Storage Service Developer Guide*.

5. Select a region for your bucket. By default, Amazon S3 creates buckets in the US Standard region. We recommend that you choose a region close to you to optimize latency, minimize costs, or to address regulatory requirements.
6. Click **Create**.
7. Select your bucket in the **Buckets** pane, and click **Upload**.
8. On the **Upload - Select Files** page, click **Add Files**, and add the following files:

   - jwplayer.flash.swf
   - jwplayer.html5.js
   - jwplayer.js
   - Your .mp4 or .flv media file.

9. Grant public read permissions for the files that you added in the previous step.

    a. Click **Set Details**.

    b. On the **Set Details** page, click **Set Permissions**.

    c. On the **Set Permissions** page, click **Make everything public**.

10. Click **Start Upload**.

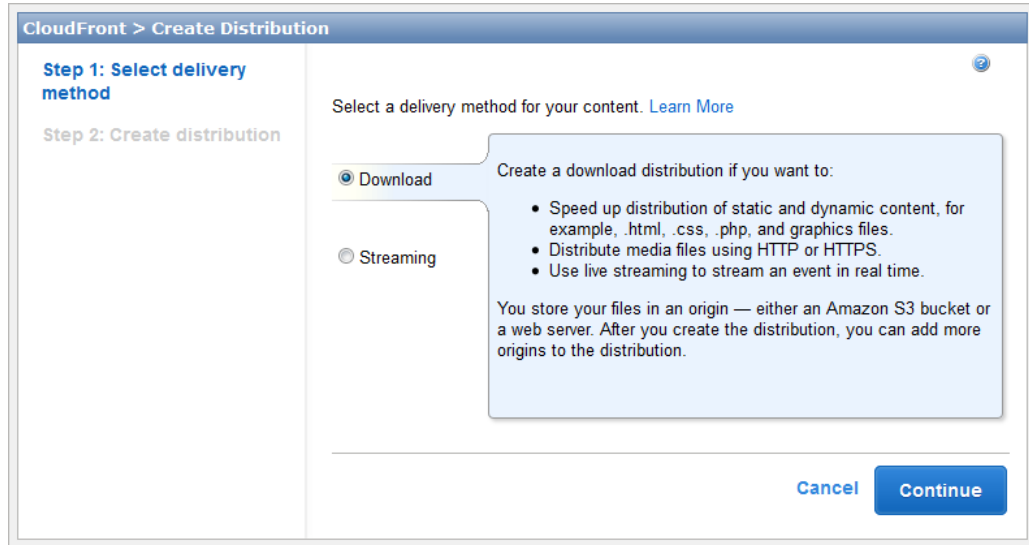# Creating CloudFront Download and Streaming Distributions

To configure CloudFront to stream a media file, you need a CloudFront download distribution for the JW Player files and an HTML file, and a streaming distribution for the media file. Perform the following two procedures to create a download distribution and a streaming distribution.

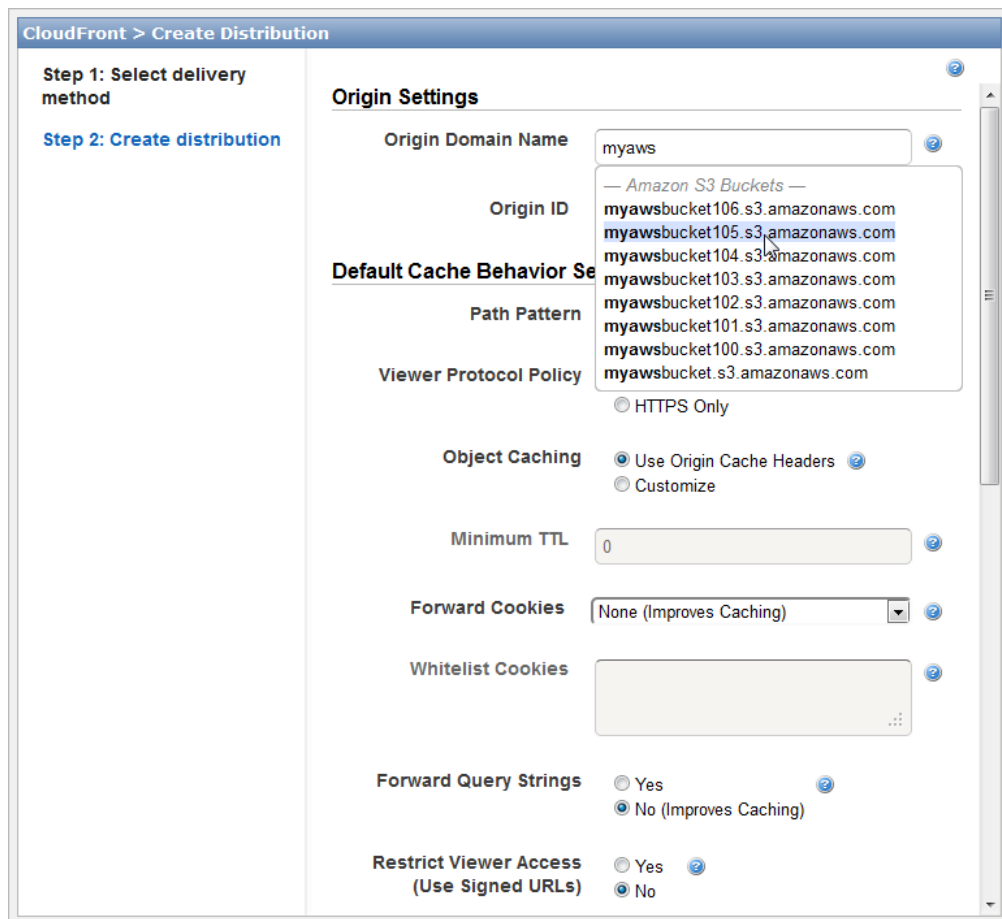**To create a CloudFront download distribution for your JW Player files**

1. Open the Amazon CloudFront console at https://console.aws.amazon.com/cloudfront/.

2. Click **Create Distribution**.

3. On the first page of the **Create Distribution Wizard**, accept the default selection, **Download**, and click **Continue**.

4. On the second page of the wizard, click in the **Origin Domain Name** field, and select the Amazon S3 bucket that you created in the procedure To upload media and JW Player files to an Amazon S3 bucket (p. 244). If you have a lot of Amazon S3 buckets, you can type the first few characters of the bucket name to filter the list.

5. Accept the default values for the remaining fields, and click **Create Distribution**.

6. After CloudFront creates your distribution, the value of the **Status** column for your distribution will change from **InProgress** to **Deployed**. This should take less than 15 minutes.

   The domain name that CloudFront assigns to your distribution appears in the list of distributions. The domain name also appears on the Distribution Settings page for a selected distribution.)

To create a CloudFront streaming distribution for your media file

1. In the CloudFront console, click **Create Distribution**.

2. In the **Create Distribution Wizard**, click **Streaming**, and click **Continue**.

3. On the second page of the wizard, click in the **Origin Domain Name** field, and select the Amazon S3 bucket that you created in the procedure To upload media and JW Player files to an Amazon S3 bucket (p. 244). If you have a lot of Amazon S3 buckets, you can type the first few characters of the bucket name to filter the list.

4. Accept the default values for the remaining fields on the **Create Distribution** page, and click **Create Distribution**.

5. After CloudFront creates your distribution, the value of the **Status** column for your distribution will change from **InProgress** to **Deployed**. This should take less than 15 minutes.

The domain name that CloudFront assigns to your distribution appears in the list of distributions. The domain name also appears on the Distribution Settings page for a selected distribution.

Embedding Video in a Web Page

The following example shows you how to embed a video in a web page using the download and streaming distributions that you created in Creating CloudFront Download and Streaming Distributions (p. 245). Perform the following steps:

1. Copy the HTML code below, and paste it into a text editor.
2. Review the comments in the HTML file, and replace the following placeholders with the applicable values:

    -   DOWNLOAD-DISTRIBUTION-DOMAIN-NAME
    -   STREAMING-DISTRIBUTION-DOMAIN-NAME
    -   VIDEO-FILE-NAME

3. Save the file with a .html filename extension, for example, `jwplayer-example.html`.

```
<HTML>
<HEAD>
<TITLE>Amazon CloudFront Streaming with Longtail Video JW Player 6</TITLE>

<!--  This HTML file plays an .mp4 or .flv media file using JW Player 6.
The following code is from the Longtail Video Setup Wizard at
http://www.longtailvideo.com/support/jw-player-setup-wizard.
-->

<!-- Call the JW Player JavaScript file, jwplayer.js.
Replace DOWNLOAD-DISTRIBUTION-DOMAIN-NAME with the domain name of your
CloudFront download distribution, for example, d1234.cloudfront.net
(begins with "d"). This causes a browser to download the JW Player file
before streaming begins.
-->

<script type='text/javascript' src='https://DOWNLOAD-DISTRIBUTION-DOMAIN-
NAME/jwplayer.js'></script>

</HEAD>

<BODY>
<H1>This video is streamed by CloudFront and played by JW Player 6.</H1>

<!-- Replace STREAMING-DISTRIBUTION-DOMAIN-NAME with the domain name of your
streaming distribution, for example, s5678.cloudfront.net (begins with "s").

Replace VIDEO-FILE-NAME with the name of your .mp4 or .flv video file,
including the .mp4 or .flv filename extension. For example, if you uploaded
my-vacation.mp4, enter my-vacation.mp4.
-->

<div id='mediaplayer'></div>
<script type="text/javascript">
    jwplayer('mediaplayer').setup({
```

```
    file: "rtmp://STREAMING-DISTRIBUTION-DOMAIN-NAME/cfx/st/VIDEO-FILE-NAME",

    width: "720",
    height: "480"
  });
</script>

</BODY>
</HTML>
```

Uploading the HTML File and Playing the Video

To play the video using the HTML file that you created in Embedding Video in a Web Page (p. 248), upload the file to your Amazon S3 bucket, and use the URL for your CloudFront distribution.

To upload the HTML file and play the video

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Select your bucket, and click **Upload**.
3. On the **Upload - Select Files** page, click **Add Files**, and add your HTML file.
4. Grant public read permissions for the HTML file that you added in the previous step.

    a. Click **Set Details**.
    b. On the **Set Details** page, click **Set Permissions**.
    c. On the **Set Permissions** page, click **Make everything public**.

5. Click **Start Upload**.
6. To play the video, enter the following URL in a web browser:

    ```
    http://domain name of your CloudFront distribution/your HTML file name
    ```

Amazon CloudFront Resources

The following table lists related resources that you'll find useful as you work with CloudFront.

Resource	Description
Amazon CloudFront API Reference	The *Amazon CloudFront 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.
Amazon CloudFront product information	The primary web page for information about Amazon CloudFront.

Resource	Description
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.

Where Do I Go from Here?

Although fairly simple to use, CloudFront is rich in functionality. There are a variety of resources you can use to learn more about CloudFront. In addition to the Amazon CloudFront documentation, resources include AWS SDKs, feature summaries and links to third-party applications that work with CloudFront. The list below includes a few of these resources that you might find helpful.

Topics

Amazon CloudFront SDKs

AWS offers SDKs that let you access CloudFront programmatically.

- AWS SDK for Java
- AWS SDK for .NET
- AWS SDK for PHP
- AWS SDK for Ruby

Using CloudFront Logging

The following AWS blog posts discuss enhancements to CloudFront logging as well as some ways to analyze access logs.

AWS Blog: Amazon CloudFront Request Logging (for content delivered via HTTP)

AWS Blog: Amazon CloudFront Now Supports Streaming Access Logs (for content delivered via RTMP)

AWS Blog: Enhanced CloudFront Logs, Now With Query Strings

Setting a Default Root Object

CloudBerry Lab: How to set CloudFront Default Object with CloudBerry S3 Explorer

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/

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

Distributing Streaming Media

AWS Articles & Tutorials:

- Configuring Amazon CloudFront Streaming Using Adobe's Flash Builder
- Configuring Amazon CloudFront Streaming Using Flowplayer
- Configuring Amazon CloudFront Streaming Using JW Player

StreamingMedia.com: How To Get Started With Amazon CloudFront Streaming

Ioncannon.net:

- iPhone Windowed HTTP Live Streaming Using Amazon S3 and CloudFront Proof of Concept
- HTTP Live Video Stream Segmenter and Distributor
- iPhone Windowed HTTP Live Streaming Server

Flowplayer.org: Bandwidth detection: Make sure you reach your entire audience with good quality

LongtailVideo.com (JW Player):

- Using Amazon Web Services
- Using CloudFront

Improving Web Site Performance

AWS Blog: Improving Global Application Performance

Creating Secure Connections Using HTTPS

AWS Blog: Amazon CloudFront: HTTPS Access, Another Edge Location, Price Reduction

Tools for Configuring Private Content

In addition to the methods provided by CloudFront, the following third-party tools provide web forms for configuring your distribution for private content. Some of the tools also provide web forms for creating signed URLs.

- **CloudBuddy:** Supports configuring a distribution for private content and supports creating signed URLs.

  For more information about using CloudBuddy for CloudFront private content, go to 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:** Supports configuring a distribution for private content

  For information about using Bucket Explorer for CloudFront private content, go to How to Create a Private Distribution on a Bucket.
- **CloudBerry:** Supports configuring a distribution for private content and supports creating signed URLs.

  For information about using CloudBerry for CloudFront private content, go to How to Configure Private Content for CloudFront Streaming with CloudBerry.

For more information about private content, see the AWS Blog: New Amazon CloudFront Feature: Private Content.

Using Custom Origins

AWS Blog: New Amazon CloudFront Feature: Custom Origins

Amazon CloudFront Third-party Tools Overview

AWS Blog: CloudFront Management Tool Roundup

Using CloudFront with a Content Management System

Drupal

- Drupal.org: CloudFront Installation
- DrupalModules.com: CloudFront Drupal Module

Sitecore

- NTT Data Advisory Service: AWS CloudFront Sitecore Integration

WordPress

- om4.com: Using Amazon CloudFront with WordPress and WordPress MU
- WordPress.org: W3 Total Cache
- WordPress.org: Simple Amazon S3 Upload Form
- WordPress.org: OSSDL CDN Off-linker
- WordPress.org: My CDN
- Inquisiter.com: Amazon CloudFront CDN with a WordPress Blog

Document History

The following table describes the important changes to the documentation since the last release of CloudFront.

- **API Version:** 2012-07-01
- **Latest documentation date:** September 5, 2012

Change	Description	Date Changed
New Features	This release of CloudFront introduces the following features: • **Fields for private content in the AWS Management Console:** Settings for private content, which previously could be configured or changed only using the CloudFront API, can now be configured or changed in the AWS Management Console. This includes settings for origin access identities and trusted signers. In addition, the documentation about private content was reorganized and clarified. For more information, see Serving Private Content through CloudFront (p. 87). • **Improvements to the AWS Management Console:** Wizards and dialog boxes in the AWS Management Console have been resized to simplify viewing on tablet computers without compromising the appearance for other viewers. In addition, the number of pages in the Create Distribution wizard was reduced to simplify the process of creating a new distribution.	September 27, 2012

Change	Description	Date Changed
New Features	This release of CloudFront introduces the following features: • **Access log improvements for download distributions:** For download distributions, CloudFront access logs now include fields for: • The cookie header in each viewer request, including name-value pairs and attributes. This field is optional. • The result type of a request (for example, Hit, RefreshHit, or Miss). • An identifier that uniquely identifies each request (the CloudFront request ID). For more information, see Download Distribution File Format (p. 147). • **Cookie support for download distributions:** You can now choose whether you want CloudFront to forward cookies and the associated cookie attributes to your origin. If so, you can also choose whether to forward all cookies or just a selected list of cookies. For more information, see How CloudFront Forwards, Caches, and Logs Cookies (p. 62). • **Price classes for download and streaming distributions:** You can now choose a price class that corresponds with the maximum price that you want to pay for CloudFront service. If you're willing to accept higher latency for your viewers in some geographic regions in return for lower cost, you can choose a price class that doesn't include all CloudFront regions. For more information, see Choosing the Price Class for a CloudFront Distribution (p. 55).	September 5, 2012
New Features	This release of CloudFront introduces the following features: • You can now invalidate objects using the CloudFront console. For more information, see Invalidating Objects (Download Distributions Only) (p. 67). • The CloudFront console was updated to better support viewing on tablet devices.	June 22, 2012

Change	Description	Date Changed
New Features	This release of CloudFront introduces the following features for download distributions: • You can forward query strings to your origin. For more information, see How CloudFront Forwards, Caches, and Logs Query String Parameters (p. 61). • You can specify up to 10 origins. For more information, see Values that You Specify When You Create or Update a Download Distribution (p. 32). • You can specify path patterns. For more information, see Values that You Specify When You Create or Update a Download Distribution (p. 32). In addition, the CloudFront console has been updated. For more information, see Creating Download Distributions (p. 16) and Creating Streaming Distributions (p. 22). The *Amazon CloudFront Getting Started Guide* was merged into the *Amazon CloudFront Developer Guide*, and the *Amazon CloudFront Developer Guide* was reorganized to enhance usability.	May 13, 2012
Updated Documentation	The documentation about working with objects was reorganized and clarified. For the revised documentation, see Working with Objects (p. 58).	April 4, 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. 188).	April 1, 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. 167).	March 29, 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: • CloudFront product page • "Caching Duration and Minimum TTL" at Request and Response Behavior for Amazon S3 Origins (p. 78) • "Caching Duration and Minimum TTL" at Request and Response Behavior, and Supported HTTP Status Codes for Custom Origins (p. 81) • The `CachingBehavior` element in the DistributionConfig Complex Type.	March 15, 2012
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. 167) chapter in this guide.	February 2, 2012

Change	Description	Date Changed
New Feature	This release of 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 Download Distributions (p. 16) • Using an HTTPS Connection to Access Your Objects (p. 136) • Specifying a Default Root Object (Download Distributions Only) (p. 72)	April 27, 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 Using IAM to Control Access to CloudFront Resources (p. 139) in the *Amazon CloudFront Developer Guide*.	March 10, 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 Download Distributions (p. 16) in the *Amazon CloudFront Developer Guide*.	November 9, 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 (Download Distributions Only) (p. 67) in the *Amazon CloudFront Developer Guide*.	August 31, 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 (Download Distributions Only) (p. 72).	August 5, 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. 147).	July 14, 2010
New Feature	Added support for secure connections using HTTPS. For more information, see Using an HTTPS Connection to Access Your Objects (p. 136).	June 7, 2010
New Feature	Added logging for streaming content. For more information, see Streaming Distribution Log File Format (p. 149).	May 13, 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. 65).	April 13, 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 Serving Private Content through CloudFront (p. 87).	March 28, 2010
New Feature	Added feature to deliver streaming content over a Real-Time Messaging Protocol (RTMP) connection. For more information, see Creating Streaming Distributions (p. 22).	December 15, 2009

Change	Description	Date Changed
New Feature	Added feature to restrict access to your content delivered over HTTP. For more information, see Serving Private Content through CloudFront (p. 87).	November 11, 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.	November 11, 2009