

## Allissa Hertz

### Writing Sample

This writing sample is an excerpt from a how-to document I wrote for Wowza Streaming Engine. There are additional steps to the article that have been left out in this version. This represents original writing. I was the only editor for this article. I gathered the information for this article on a call with an SME. All code samples were provided by the SME.

I was asked to produce this article with a quick turnaround because it is a workaround for an issue that was causing support calls. I met with the SME to understand the process in the morning and then I published it in the evening of the next day.

# Send Apple HLS content to Amazon S3 using Wowza REST APIs

This article explains how to create a Fastly stream target with a push connection to distribute your live streams using Wowza CDN on Fastly. You will need an Amazon AWS account and a Wowza Streaming Cloud account for this workflow.

Alternatively, you may configure your connection to for a pull-based workflow where the destination pulls the stream data from the server.

Unless there is a specific need for a push-based workflow, it's strongly recommended to use a pull-based workflow instead. See [Migrate to Wowza CDN on Fastly](#) to learn more about the advantages and disadvantages of the push and pull workflows.

To configure your stream target using a pull connection, see [Stream to Wowza CDN from Wowza Streaming Engine using Wowza REST APIs](#).

## Before you start

You should have access to the following items:

- A valid **Wowza Streaming Engine license and a Wowza CDN subscription**. Contact [sales@wowza.com](mailto:sales@wowza.com) for more information.
- An Amazon Web Service account with create, read, and write access to S3 storage using access key security credentials. See [AWS Free Tier](#) to create a free account.

## 1. Configure a bucket in AWS S3

**Note:** See [Creating a bucket](#) for instructions on how to create a bucket in AWS.

You'll need to configure some settings to allow streams to be played back from your S3 bucket.

### Configure the following:

- **ACLs** must be **enabled**. For best results, the Object Ownership should be set to the Object Writer.

### Object Ownership [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

- ACLs disabled (recommended)**  
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

- ACLs enabled**  
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

#### Object Ownership

- Bucket owner preferred**  
If new objects written to this bucket specify the bucket-owner-full-control canned ACL, they are owned by the bucket owner. Otherwise, they are owned by the object writer.
- Object writer**  
The object writer remains the object owner.

- **Public Access must be enabled** for objects in the bucket.

### Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#) 

- Block all public access**  
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.
- Block public access to buckets and objects granted through new access control lists (ACLs)**  
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
  - Block public access to buckets and objects granted through any access control lists (ACLs)**  
S3 will ignore all ACLs that grant public access to buckets and objects.
  - Block public access to buckets and objects granted through new public bucket or access point policies**  
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
  - Block public and cross-account access to buckets and objects through any public bucket or access point policies**  
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.



**Turning off block all public access might result in this bucket and the objects within becoming public**  
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

- I acknowledge that the current settings might result in this bucket and the objects within becoming public.

- For testing direct playback from the bucket, you may need to enable Cross-origin resource sharing (CORS). After the bucket is created, select the **Permissions** tab for the bucket and add the following **CORS configuration**.

**Note:** CORS is only required for direct playback from the bucket. It's not required for playback via the Fastly Custom Stream Target.

```
[
  {
    "AllowedHeaders": [],
    "AllowedMethods": [
      "GET"
    ],
    "AllowedOrigins": [
      "*"
    ],
    "ExposeHeaders": []
  }
]
```

Once your bucket is created, you'll need to record the following information from your amazon account:

- **Bucket name**, for example *my-s3-bucket*
- **Bucket Region**, for example *us-west-1*
- **AWS Bucket URL**. This will be used with the Fastly Custom Stream Target. The format for the AWS Bucket URL is *https://[bucket-name].s3.[region].amazonaws.com/*

For example, <https://my-s3-bucket.s3.us-west-1.amazonaws.com/>

**Note:** For buckets in the us-east-1 region, AWS doesn't require the region in the URL ([https://\[bucket-name\].s3.amazonaws.com/](https://[bucket-name].s3.amazonaws.com/)). This is a legacy format AWS URL. It won't work with Fastly Custom Stream Targets. Instead, you need to use the regionalised S3 URL for us-east-1, which is [https://\[bucket-name\].s3.us-east-1.amazonaws.com](https://[bucket-name].s3.us-east-1.amazonaws.com/)

You'll also need to record your IAM credentials. These are set per user in AWS. See [Create an IAM admin and user group in AWS](#) for more information.

Record the following:

- **Access ID**, for example *AKIAI6234VXXREN3KWJQ*
- **Secret Access Key**, for example *y1PFFPOEwSrUfvvvdalA1qs9sFDM7+QzQTMHoqP7*

## 2. Create a stream target in Wowza Streaming Engine to send the stream to AWS S3

1. Once you have created your AWS S3 bucket and recorded your credentials, you'll need to create a stream target in Wowza Streaming Engine by sending a *POST* request to the */v2/servers/\_defaultServer\_/vhosts/\_defaultVHost\_/applications/live/publish/mapentries/WSC\_via\_S3* endpoint.

You can use the following sample request, making sure to:

- Set *http.relativePlaylists* to **true**. It is set to false by default.
- Set *entryName* to a unique name for your stream target.  
For Adaptive Bitrate streams, set *adaptiveGroup* to the same value for each stream entry.
- Set *cloudstorage.string.bucketName* to the **Bucket Name** provided by AWS in Step 1.
- Change any additional values to be unique to your stream target.
- Before saving the file, use a JSON Validator, such as <https://jsonlint.com>, to validate the map entries. Copy the complete text after the = sign into the validator to check the syntax.

### Sample Request

```
curl -X POST \  
-H 'Accept:application/json; charset=utf-8' \  
-H 'Content-Type:application/json; charset=utf-8' \  

```

```
http://localhost:8087/v2/servers/_defaultServer_/vhosts/_defaultVHost_/applications/1
ive/publish/mapentries/WSC_via_S3_720p \
-d '
{
  "entryName": "WSC_via_S3_720p",
  "sourceStreamName": "stream_name",
  "streamName": "stream_name",
  "profile": "cupertino-cloudstorage",
  "debugLog": false,
  "extraOptions": {
    "cloudstorage.provider": "S3",
    "cloudstorage.string.provider": "S3",
    "cloudstorage.string.bucketName": "my-s3-bucket",
    "cloudstorage.string.region": "us-west-1",
    "cloudstorage.string.accessID": "AKIAI6IO5VXX-EXAMPLE",
    "cloudstorage.string.secretAccessKey": "y1PFFPOEwSrUfWyha1A1qs9sFDM7+QzQ-exam
ple",
    "cloudstorage.transportDebug": false,
    "http.relativePlaylists": true
  }
}
```

## Sample Response

```
{
  "success": true,
  "message": "Entry (WSC via S3 - source) saved successfully",
  "data": null
}
```

2. Copy the **URL** for the playlist from your S3 bucket, for example *<https://my-s3-bucket.s3-us-west-1.amazonaws.com/myStream/playlist.m3u8>*

**Note:** For buckets in the [us-east-1](#) region, this URL will be the legacy format and needs to be changed to include the region.

3. Test the playback using the **URL** for your playlist in your player.