# STREAMING VIDEO TO AWS ELEMENTAL MEDIACONNECT USING ZIXI SDK – PART 2

# CONTENTS


INTRODUCTION ..... 3

PRE-REQUISITES ..... 3

COST OF PROVISIONING RESOURCES USED IN THIS POST ..... 3

STEP 1: CONFIGURE ENCRYPTION AND PERMISSIONS ..... 3

STEP 2: CREATE AWS IDENTITY AND ACCESS MANAGEMENT POLICIES AND IAM USER FOR AWS ELEMENTAL LIVE ..... 3

STEP 3: CONFIGURE MEDIACONNECT FLOW INPUT AND OUTPUT ..... 4

STEP 4: CHOOSE AN ENCODER TYPE (AWS ELEMENTAL LIVE OR FFMPEG) ..... 4

STEP 5: AWS ELEMENTAL LIVE – CONFIGURE EVENT INPUTS ..... 4

STEP 6: CONFIGURE AWS ELEMENTAL LIVE EVENT OUTPUT ..... 5

STEP 7: STREAM VOD ASSETS (.MP4 FILES) ..... 6

STEP 8: FFMPEG – CREATE A MEDIACONNECT FLOW ..... 7

STEP 9: COMPILE FFMEG TO SUPPORT FORWARD ERROR CORRECTION (FEC) AND INSTALL FFMPEG ..... 7

STEP 10: INSTALL ZIXI-MEDIACONNECT AND CONFIGURE AS FEEDER ..... 8

STEP 11: STREAM VOD ASSETS (.MP4 FILES) ..... 9

STEP 12: VIEW VOD ASSET ON ANDROID OR IOS DEVICE ..... 10

STEP 13: CLEAN UP ..... 10

CONCLUSION ..... 11

RESOURCES ..... 11

## Introduction

Video-on-demand (VOD) assets can be streamed to end-user mobile devices by using a local encoder, AWS Elemental MediaConnect (EMX) service, and Zixi Player.

In Part 1 of this blog series, we demonstrated how to install and configure Open Broadcaster Software (OBS) Studio and Zixi SDK for AWS Elemental MediaConnect (Zixi-MediaConnect) Feeder on a local system, create an AWS Elemental MediaConnect flow, and stream a local VOD asset to end-user mobile devices.

In this document, we demonstrate how to configure a local encoder (either AWS Elemental Live or FFmpeg) and AWS Elemental MediaConnect to encode and stream VOD assets from Amazon Simple Storage Service (Amazon S3) or a local system to end-user mobile devices.

A description of Zixi-MediaConnect Feeder/Receiver, supported protocols, latency considerations, and limitations are available in Part 1 of this blog.

## Pre-requisites

This post requires access to an encoder (either an AWS Elemental Live encoder or a host system such as Ubuntu Focal Fossa, capable of running FFmpeg).

## Cost of provisioning resources used in this post

- There is no charge for FFmpeg, Zixi-MediaConnect Feeder/Receiver, or the Zixi Player app.
- MediaConnect is billed at an hourly rate for each running flow, plus either a per-gigabyte charge for data transferred using the flow or, with reserved outbound bandwidth pricing, a per-hour charge based on bandwidth for video sent to the internet. A flow that is in standby mode is not an active resource, and does not incur active resource costs even if it has outputs or entitlements configured. See https://aws.amazon.com/mediaconnect/pricing/ for more information.
- AWS Secrets Manager comes with a 30-day free trial. After the trial is over, Secrets Manager is billed $0.40 per secret per month. For secrets that are stored for less than a month, the price is prorated (based on the number of hours). See https://aws.amazon.com/secrets-manager/pricing/ for more information.
- To keep charges to a minimum, be sure to clean up resources once you are finished testing by following the tasks in Step 13 of this post.

## Step 1: Configure encryption and permissions

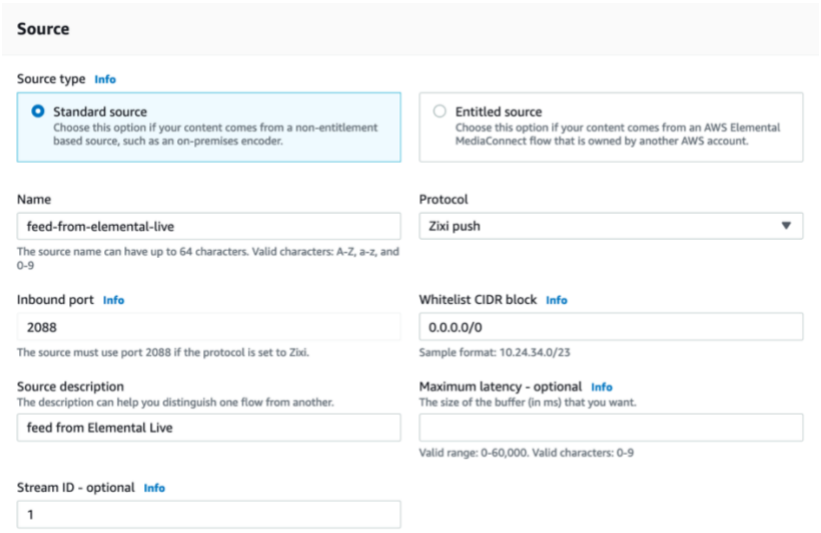
Perform the tasks in Step 1 of the Part 1 blog post. These tasks create a secret and also create a policy and role allowing MediaConnect to connect to AWS Secrets Manager.

## Step 2: Create AWS Identity and Access Management policies and IAM user for AWS Elemental Live

Execute tasks in Step A in this AWS Elemental Live setup guide to create two new IAM policies and a new IAM user. Note: we have already covered the tasks in Step B (encryption setup) of the AWS Elemental Live setup guide. Steps C and D (MediaConnect flow and output setup) are covered in the following.

## Step 3: Configure MediaConnect flow input and output

a. Perform the tasks in Step 2 of the Part 1 blog post, except use the following **Source** values:
   **Source type**: Standard source
   **Name**: feed-from-elemental-live
   **Protocol**: Zixi push
   **Inbound port**: 2088 (default)
   **Source description**: feed from AWS Elemental Live
   **Stream ID**: 1




b. Perform the remaining tasks of Step 2 in the Part 1 blog post to configure Decryption settings and flow Output and start the MediaConnect flow.

## Step 4: Choose an encoder type (AWS Elemental Live or FFMpeg)

Decide if you want to proceed using AWS Elemental Live or FFmpeg as your encoder. If you use AWS Elemental Live as your encoder, continue with Step 5. If you would like to use FFmpeg as your encoder, skip directly to Step 8.

## Step 5: AWS Elemental Live – configure Event inputs

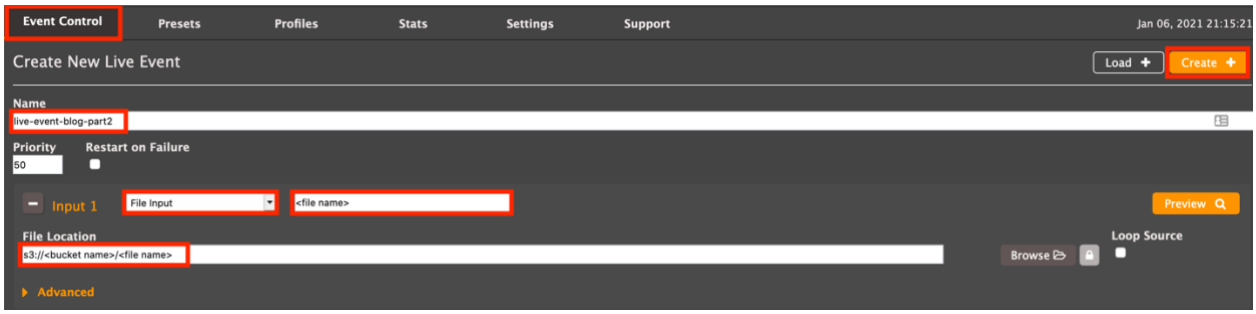
*Skip to Step 8 if you aren't using AWS Elemental Live*

a. Obtain two or more .mp4 files to use for this demo and upload to a new or existing Amazon S3 bucket in your AWS account.
b. Log in to the AWS Elemental Live encoder.

Our first step is to configure a new Live Event.

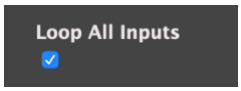
c. Click **Event Control**, and Click **Create**
d. Fill in the values as shown:
   **Name**: live-event-blog-part2
   **Priority**: <leave default>
   **Restart on Failure**: <leave default>
   **Input 1:** choose **File Input**

**Name:** choose any name – this name will be used to help you identify the file
**File Location**: type or paste the Amazon S3 ARI for the file in the format s3://<bucket name>/<file name>




e. Click the lock icon to supply AWS IAM credentials
**User name**: Paste the AWS access key ID from Step 2
**Password**: Paste the Secret access key from Step 2

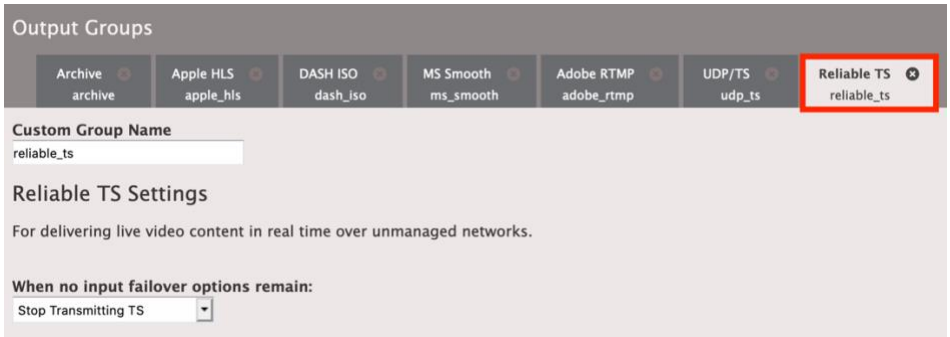

f. Click **Add Input**
g. Configure File input, (name), File Location, and credentials for your remaining .mp4 files in Amazon S3
h. When you finish creating the last input, check the **Loop All Inputs** box.




## Step 6: Configure AWS Elemental Live Event output

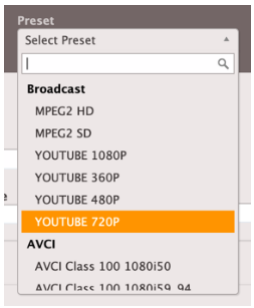
*Skip to Step 8 if you aren't using AWS Elemental Live*

a. Scroll down to **Output Groups** and select **Reliable TS**
Fill in the values as shown:
**Custom Group Name:** reliable_ts (default)
**When no input failover options remain:** Stop Transmitting TS (default)

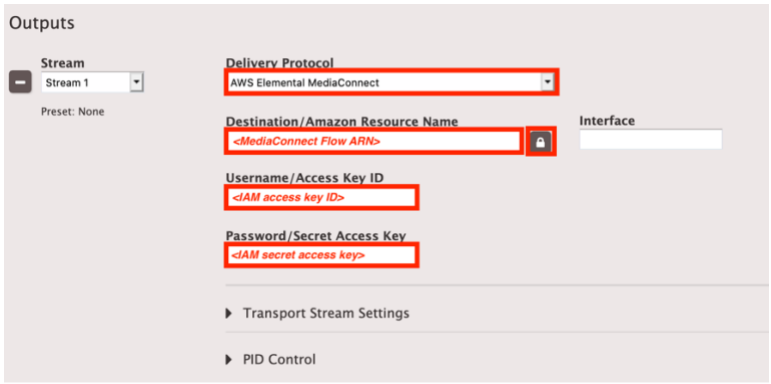



b. Click **Add Output.**


c. In the **Preset** pull-down, choose **YOUTUBE 720P**




d. Fill in the values as shown:
**Delivery Protocol:** AWS Elemental MediaConnect
**Destination:** MediaConnect flow ARN from Step 3
**User name**: paste the IAM access key ID from Step 2
**Password**: paste the IAM secret access key from Step 2



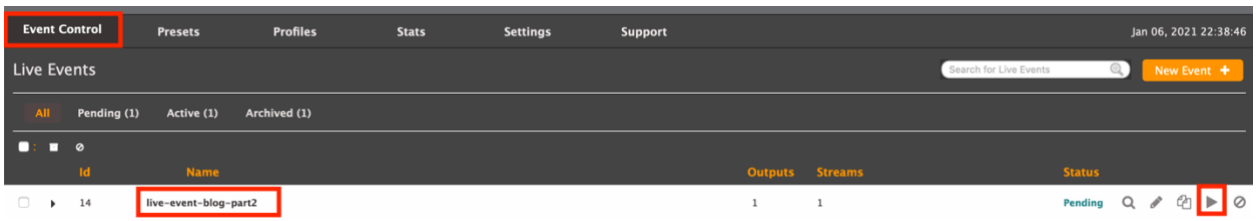

e. Click **Create**


## Step 7: Stream VOD assets (.mp4 files)

*Skip to Step 8 if you aren't using AWS Elemental Live*

We can now begin streaming our VOD assets from AWS Elemental Live to MediaConnect.

a. In AWS Elemental Live, click Event Control, locate the Event you just created, and click the **Start arrow icon** (far right).



AWS Elemental Live encoder setup is complete. AWS Elemental Live now streams the .mp4 files (in a continuous, encrypted loop) to MediaConnect.

b. Skip over Steps 8-11 and go directly to Step 12 to view the output on your mobile device.

## Step 8: FFMpeg – create a MediaConnect flow

*Skip to Step 12 if you aren't using FFMpeg*

FFMpeg streams to MediaConnect. To create a MediaConnect flow, perform the tasks in Step 2 of the Part 1 blog post.

## Step 9: Compile FFmeg to support forward error correction (FEC) and install FFmpeg

*Skip to Step 12 if you aren't using FFMpeg*

a. Log into Ubuntu system and open a terminal window.

FFmpeg is open-source software distributed under the LGPL2.1 license.

b. If you have a version of FFmpeg already installed, uninstall it:

[Gnu Bash]

```
$ sudo apt-get remove ffmpeg
```

c. Now you must compile FFmpeg to include FEC capability:

[Gnu Bash]

```
$ cd ~
$ sudo apt -y update
$ sudo apt -y upgrade
$ wget https://ffmpeg.org/releases/ffmpeg-snapshot.tar.bz2
$ tar -jxvf ffmpeg-snapshot.tar.bz2
$ cd ffmpeg/
$ sudo apt -y install autoconf automake build-essential cmake git-core libass-dev libfreetype6-dev libgnutls28-dev libsdl2-dev libtool libva-dev libvdpau-dev libvorbis-dev libxcb1-dev libxcb-shm0-dev libxcb-xfixes0-dev pkg-config texinfo wget yasm zlib1g-dev nasm libx264-dev libx265-dev libnuma-dev libvpx-dev libfdk-aac-dev libmp3lame-dev libopus-dev
$ sudo ./configure --pkg-config-flags="--static" --extra-libs="-lpthread -lm" --enable-gpl --enable-gnutls --disable-libaom --enable-libass --enable-libfdk-aac --enable-libfreetype --enable-libmp3lame --enable-libopus --disable-libsvtav1 --enable-libvorbis --enable-libvpx --enable-libx264 --enable-libx265 --enable-nonfree
$ sudo make
```

d. Install FFmpeg:

[Gnu Bash]

$ sudo make install

e. Check to see if the installation was successful:

[Gnu Bash]

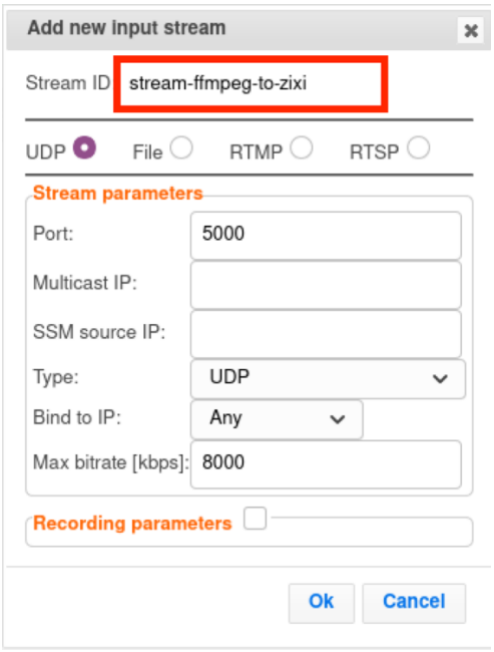
$ ffmpeg -version

You should see version information displayed.

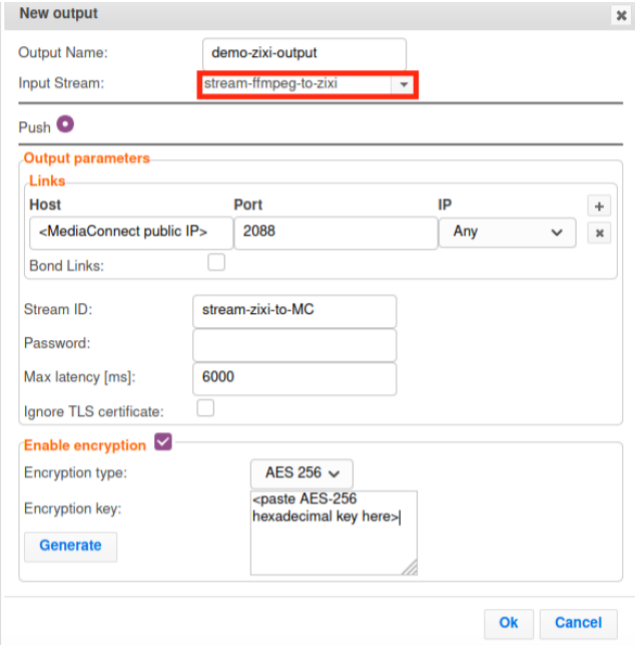
## Step 10: Install Zixi-MediaConnect and configure as Feeder

*Skip to Step 12 if you aren't using FFMpeg*

a. Perform the tasks in Step 3 of the Part 1 blog post to obtain a Zixi-MediaConnect Feeder license.
b. Perform the tasks in Step 5 (a-i) of the Part 1 Blog post Part 1 blog post to download, install, license, and configure Zixi-MediaConnect. Specify the **Stream ID**: stream-ffmpeg-to-zixi.




c. Continue performing the tasks in Step 5 (j-m) of Part 1 to configure the Zixi Output, but specify **Input Stream**: stream-ffmpeg-to-zixi




## Step 11: Stream VOD assets (.mp4 files)

*Skip to Step 12 if you aren't using FFMpeg*

a. Obtain two or more .mp4 files to use for this demo and save to your Ubuntu system home directory.
b. Open Ubuntu terminal window and create a shell script to play each video in a continuous loop:

[Gnu Bash]

```
$ vim ~/stream-videos-loop.sh
```

c. Paste the following into the file, replacing each <filename-video-1>.mp4 with the name of your .mp4 file(s):

[plain text]

```
#! /bin/bash

while true ; do
  for video in <filename-video-1>.mp4 <filename-video-2>.mp4 <filename-video-3.mp4> ; do
    ffmpeg -re -i $video -c copy -map 0 -f rtp_mpegts -fec prompeg=l=5:d=20 rtp://127.0.0.1:5000
    test $? -gt 128 && exit
  done
done
```

d. To run the script, type:

[Gnu Bash]

```
$ chmod +x stream-videos-loop.sh
$ ./stream-videos-loop.sh
```

e. Allow the script to continue running until you are finished testing on your mobile device. After testing is complete, the script can be stopped by pressing **Ctrl-C** in the terminal window.
f. In Zixi-MediaConnect, click Inputs and verify that the Zixi Input is Connected to ffmpeg.

Zixi-MediaConnect pushes the encrypted stream to MediaConnect.

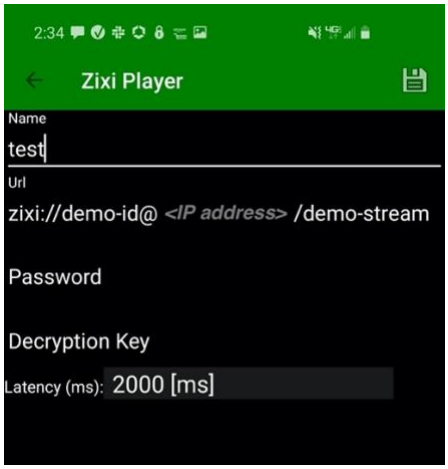
g. Continue with Step 11 to view the VOD asset on your mobile device.

## Step 12: View VOD asset on Android or iOS device

a. On your mobile device, install Zixi Player, available for Android from https://android-apk.org/com.zixi.player/35756756-zixiplayer/ or the App Store for iOS.

Now we configure Zixi Player to receive the feed from MediaConnect.

b. Open the Zixi Player app on your mobile device and fill in the values as shown:
**Name**: test
**URL**: zixi://demo-id@<MediaConnect Public Outbound IP address>/demo-stream
**Password**: <leave blank>
**Decryption key**: <leave blank>
**Latency**: default value




c. Tap **Save (disk icon)** in the upper right.
d. Tap the name of the configuration to begin playing the stream.

## Step 13: Clean up

a. If you used the FFmpeg encoder, press **Ctrl-C** in the Ubuntu terminal window to stop the stream-videos.sh script.
b. If you used an AWS Elemental Live encoder, **Stop** the "live-event-blog-part2" Live Event.
c. In the AWS Elemental MediaConnect console, **Stop** the "demo-part2" flow and then **Delete** it.
d. In the AWS Secrets Manager console, select the demo-MediaConnect-AES256 secret and click **Actions, Delete Secret**. Choose a 7-day waiting period (this is the minimum) and click **Schedule Deletion.**

## Conclusion

In this post, you used an AWS Elemental Live encoder or FFmpeg on an Ubuntu system to read VOD assets from Amazon S3 and securely stream the assets to MediaConnect. The VOD assets were consumed by Zixi Player on your mobile device. In Part 3 of this blog series, we explore streaming to Zixi-MediaConnect Receiver and OTT workflows.

## Resources

- AWS Elemental MediaConnect Source Failover
- AWS Elemental MediaConnect Zixi portal
- Services and tools for monitoring video workflows on AWS
- The evolution of the television live event ecosystem and how AWS and TAGVS are shaping the future
- Zen master and AWS Elemental MediaConnect. Scale technical operations with global visibility
- AWS Elemental MediaConnect user guide
- Content Syndication using AWS Elemental MediaConnect