Ingest and convert timed metadata with the Wowza Streaming Cloud REST API

Timed metadata allows you to add interactivity to streams broadcast through the Wowza Streaming Cloud™ service. Wowza Streaming Cloud can ingest AMF data, a type of metadata that is embedded into WOWZ or RTMP streams, from source encoders, cameras, Wowza Streaming Engine™, or Wowza GoCoder™ SDK broadcast apps. To send the metadata for playback over HLS streams, Wowza Streaming Cloud must first convert AMF data to ID3 tags.

In this article, learn how to ingest AMF data into Wowza Streaming Cloud, convert AMF data into ID3 tags using the Wowza Streaming Cloud REST API, and consider available playback options for timed metadata in HLS streams.

Ingest AMF data into a Wowza Streaming Cloud transcoder

Wowza Streaming Cloud can receive AMF metadata from an encoder or other H.264 source that supports and sends it, but AMF metadata can’t be directly injected into a transcoder in Wowza Streaming Cloud.

Wowza Streaming Engine and Wowza GoCoder SDK broadcast apps can send single bitrate streams that contain AMF metadata to a Wowza Streaming Cloud transcoder. For instructions on these workflows, see More resources.

Alternatively, Wowza Streaming Engine and Wowza GoCoder SDK broadcast apps can send streams that contain AMF data directly to origin servers for ultra low latency stream targets, bypassing Wowza Streaming Cloud transcoders. For instructions on those workflows, see More resources.

**Note:** In order for Wowza Streaming Cloud to convert AMF metadata to ID3 tags for HLS playback, it must be in the following format:

- The AMF data must be included in a top-level AMF data object (AMFDataObj). Within that AMF data object, you can include key:value pairs that include AMF data of additional types, such as string, Boolean, list,
or even a nested object.

- The **AMFDataObj** object must include two properties: (1) a key called **payload** whose value is a string of data to be converted and (2) a key called **wowzaConverter** whose value is **basic_string**. For a code example showing how to format incoming AMF data, see [Inject timed metadata using a Wowza Streaming Engine HTTP provider](#).

For additional instructions on sending AMF metadata with a third-party encoder, see the manufacturer’s documentation.

**Convert AMF data into ID3 tags**

**Create a transcoder to receive the source**

AMF data enters Wowza Streaming Cloud through an RTMP stream that flows into a transcoder. Before enabling the property that converts AMF data to ID3 tags, you need to create a transcoder if you haven’t already.

**Tips:**

- If you have already created a transcoder with associated outputs and targets in Wowza Streaming Cloud, use a GET query to fetch the details of the transcoder, including the ID of the stream target. For next steps, skip to **Update stream target properties**.

```bash
curl -X GET \
-H "wsc-api-key: ${WSC_API_KEY}" \
-H "wsc-access-key: ${WSC_ACCESS_KEY}" \
"${WSC_HOST}/api/${WSC_VERSION}/transcoders/[transcoder_id]"
```

- If you have already created a live stream in Wowza Streaming Cloud, use a GET query to fetch the details of the live stream, including the ID of the stream target. For next steps, skip to **Update stream target properties**.

```bash
curl -X GET \
-H "wsc-api-key: ${WSC_API_KEY}" \
-H "wsc-access-key: ${WSC_ACCESS_KEY}" \
"${WSC_HOST}/api/${WSC_VERSION}/live_streams/[live_stream_id]"
```

1. Create a transcoder using the Wowza Streaming Cloud REST API. Set the protocol to RTMP so that you can ingest AMF data.
Example request and response

The following request creates a transcoder that uses any RMTP-enabled encoder, such as Wowza Streaming Engine, a mobile broadcast app created with Wowza GoCoder SDK, or a third-party encoder, as the source.

```bash
curl -X POST
-H "Content-Type: application/json"
-H "wsc-api-key: ${WSC_API_KEY}"
-H "wsc-access-key: ${WSC_ACCESS_KEY}"
-d '{
    "transcoder": {
        "billing_mode": "pay_as_you_go",
        "broadcast_location": "us_west_california",
        "name": "My transcoder with AMF data",
        "protocol": "rtmp",
        "transcoder_type": "transcoded",
        "delivery_method": "push"
    }
}' "${WSC_HOST}/api/${WSC_VERSION}/transcoders"
```

The response includes information about the transcoder, including the transcoder ID, which you will need to associate the transcoder with output renditions and stream targets. The response does not contain outputs, and it should look something like this:
2. Complete the transcoder by adding output renditions and stream targets.

When you add a stream target, ensure that you use one of these types:
- A Wowza CDN on Fastly target
- A Wowza CDN on Akamai target with the provider akamai_cupertino
- A custom target with the with the provider akamai_cupertino

Note the stream target ID for the next section. For instructions, see one of the following articles, depending on whether you’re creating an adaptive bitrate or passthrough transcoder:
- Create an ABR stream and send it to a target with the Wowza Streaming Cloud REST API
- Pass a stream through the transcoder to a target with Wowza Streaming Cloud REST API

Update stream target properties

Once you have associated a stream target with an output rendition for the transcoder, or once you have a stream target ID from your live stream details, configure a stream target to enable the conversion of AMF data to ID3 tags. If you
use multiple stream targets with the transcoder, configure the property for each one.

1. Configure the `convertAMFData` property for the stream target using the stream target ID. When you set `convertAMFData` to `true`, Wowza Streaming Cloud listens for AMF data events coming from the source encoder or camera, parses the data events, maps the events to ID3 tags, and sends the ID3 tags in the HLS stream.

For additional information about stream target properties, see Set advanced properties with the Wowza Streaming Cloud REST API.

**Note:** Unlike Wowza CDN or custom HLS streams that require you to configure the `convertAMFData` stream target property, backup HLS streams associated with ultra low latency stream targets have the property enabled by default. Wowza Streaming Cloud automatically converts AMF data entering an ultra low latency stream target to ID3 tags in the backup HLS stream, and you can’t disable this behavior. See More resources for more information about this workflow.

**Example request and response**

The following request enables the `convertAMFData` property for a specified stream target.

```bash
curl -X POST \
-H "Content-Type: application/json" \
-H "wsc-api-key: ${WSC_API_KEY}" \
-H "wsc-access-key: ${WSC_ACCESS_KEY}" \
-d '{
   "property": {
      "key": "convertAMFData",
      "section": "hls",
      "value": true
   }
}' "${WSC_HOST}/api/${WSC_VERSION}/stream_targets/[stream_target_id]/properties"
```

The response looks something like this:
2. If you have started the transcoder or live stream at any point before enabling the `convertAMFData` stream target property, you must reset the transcoder or live stream for the property to take effect. This step isn’t necessary if you haven’t started the transcoder or live stream at all.

**Example requests**

**Reset a transcoder:**

```bash
curl -X PUT \
-H "wsc-api-key: ${WSC_API_KEY}" \
-H "wsc-access-key: ${WSC_ACCESS_KEY}"
"${WSC_HOST}/api/${WSC_VERSION}/transcoders/[transcoder_id]/reset"
```

**Reset a live stream:**

```bash
curl -X PUT \
-H "wsc-api-key: ${WSC_API_KEY}" \
-H "wsc-access-key: ${WSC_ACCESS_KEY}"
"${WSC_HOST}/api/${WSC_VERSION}/live_streams/[live_stream_id]/reset"
```

**Verify the metadata in the stream**

To verify that the converted AMF data is showing up as ID3 tags in an HLS stream, you can view the console log of the Wowza Streaming Cloud hosted page.

1. Set up a live stream in Wowza Streaming Cloud that uses Wowza Player for playback and includes a hosted page.
2. Set up a source that sends AMF data, and configure the associated stream target in Wowza Streaming Cloud to convert the AMF data to ID3 tags.
3. Start the source and the live stream.
4. Open the hosted page URL, play the stream, and view the console log to verify that metadata shows up in the stream.
5. Stop the live stream and the source.

**Playback options**
When a player that supports ID3 tags receives an HLS stream, it can listen for and process the metadata. Wowza™ Player and JW Player both support ID3 timed metadata.

At this time, the GoCoder SDK doesn’t pass ID3 tags to native Apple and Android player frameworks, but you can build an HLS player independently that will leverage the timed metadata to invoke an action during playback.

For information on whether and how other third-party players support metadata formatted in ID3 tags, see their documentation.

More resources

- Inject timed metadata using a Wowza Streaming Engine HTTP provider
- Send a stream from Wowza Streaming Engine to Wowza Streaming Cloud for transcoding using Wowza APIs
- Send and receive events with Wowza GoCoder SDK for iOS
- Send and receive events with Wowza GoCoder SDK for Android
- Use timed metadata with an ultra low latency stream target in Wowza Streaming Cloud
- Listen for metadata and trigger an action with the Wowza Player JavaScript API