Pass a stream through the transcoder to a target with the Wowza Streaming Cloud REST API

Originally Published on 11/08/2016 | Updated on 02/03/2020 10:40 am PST

Learn how to use the REST API to send a stream through the Wowza Streaming Cloud™ service without processing (how to create a passthrough stream) and deliver the stream to an external, third-party destination or to the Wowza CDN. Then, learn how to programmatically start and stop the transcoder.

Create a passthrough transcoder

Start by creating a transcoder that receives the stream from a source encoder but doesn’t perform any transcoding.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>billing_mode</td>
<td>string</td>
<td>The billing mode for the stream. The default is <strong>pay_as_you_go</strong>. If you have a 24x7 subscription, choose <strong>pay_as_you_go</strong> or <strong>twentyfour_seven</strong>.</td>
</tr>
<tr>
<td>broadcast_location</td>
<td>string</td>
<td>Specify the region that’s closest to where your stream originates. Valid values are <strong>asia_pacific_australia</strong>, <strong>asia_pacific_india</strong>, <strong>asia_pacific_japan</strong>, <strong>asia_pacific_taiwan</strong>, <strong>eu_belgium</strong>, <strong>eu_germany</strong>, <strong>eu_ireland</strong>, <strong>south_america_brazil</strong>, <strong>us_central_iowa</strong>, <strong>us_east_s_carolina</strong>, <strong>us_east_virginia</strong>, <strong>us_west_california</strong>, and <strong>us_west_oregon</strong>.</td>
</tr>
</tbody>
</table>
### notes:
- **asia_pacific_taiwan**, **eu_belgium**, **us_central_iowa**, and **us_east_s_carolina** don’t allocate dedicated GPU resources to 4K, 24x7 streams. As a result, running 4K streams in the 24x7 billing mode at these locations is not recommended.
- Region availability depends on your Wowza Streaming Cloud plan provider. **asia_pacific_s_korea** and **eu_ireland** broadcast locations are available to direct Wowza subscribers only.

### delivery_method

*string*

The method you’re using to deliver the source stream to the transcoder. Use one of the following valid values:

- **pull** – (Default) Instructs Wowza Streaming Cloud to pull the stream from the RTMP or RTSP source.
- **push** – Instructs the RTMP or RTSP source to push the stream directly to Wowza Streaming Cloud.

### name

*string*

The name of the transcoder. Enter an alphanumeric string that is short (maximum 200 characters) and descriptive, for example, **MyPassthruTranscoder**.

### protocol

*string*

The transport protocol you’re using to send the encoded stream to the transcoder. Valid values are **rtmp**, **rtsp**, **srt**, **udp**, or **webrtc**.

### source_url

*string*

Required for RTMP and RTSP pull connections. Enter the source encoder’s web address, without the preceding protocol or trailing slash (/).
**transcoder_type** | **string** | Specify the value **passthrough**.

**Note:** For information on other transcoder parameters, see the Wowza Streaming Cloud REST API Reference Documentation.

---

**Example request and response**

Create a passthrough transcoder:

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

The command creates a passthrough transcoder with an *id* parameter but no outputs ("outputs": []). The details of the configured transcoder are listed in the response, which should look something like this:
Related requests

View the details of a configured transcoder:

```
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]"
```

Update a transcoder's configuration:
Create an output for the transcoder

Next, define a passthrough output rendition that uses the source encoder’s settings.

**Passthrough rendition parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>passthrough_audio</td>
<td>Boolean</td>
<td>Specify true to create the output from the highest bitrate received by the transcoder from the encoder.</td>
</tr>
<tr>
<td>passthrough_video</td>
<td>Boolean</td>
<td>Specify true to create the output from the highest bitrate received by the transcoder from the encoder.</td>
</tr>
<tr>
<td>stream_format</td>
<td>string</td>
<td>The contents of the stream. Valid values are audiovideo (both audio and video), videoonly, or audioonly.</td>
</tr>
<tr>
<td>transcoder_id</td>
<td>string</td>
<td>The unique alphanumeric string that identifies the transcoder to which you want to add the output. You can find the ID in the details of the transcoder you just created.</td>
</tr>
</tbody>
</table>

**Note:** Additional parameters are available for outputs, but they don’t apply to passthrough workflows. For information on other output parameters, see the

```
curl -X PATCH  
-H "Content-Type: application/json"  
-H "wsc-api-key: ${WSC_API_KEY}"  
-H "wsc-access-key: ${WSC_ACCESS_KEY}"  
-d '{  
    "transcoder": {  
        "name": "MyDifferentTranscoderName"  
    }  
}' "${WSC_HOST}/api/${WSC_VERSION}/transcoders/[[transcoder_id]]"
```
Example request and response
Create a passthrough output:

curl -X POST \
-H "Content-Type: application/json" \
-H "wsc-api-key: ${WSC_API_KEY}" \
-H "wsc-access-key: ${WSC_ACCESS_KEY}" \
-d '{
    "output": {
        "passthrough_audio": true,
        "passthrough_video": true,
        "stream_format": "audiovideo"
    }
}' "${WSC_HOST}/api/${WSC_VERSION}/transcoders/[transcoder_id]/outputs"

The command creates the output with an id parameter but no targets ("targets": []). The details of the configured output are listed in the response, which should look something like this:

```
{
    "output": {
        "bitrate_audio": 0,
        "bitrate_video": 0,
        "created_at": "2015-07-28T11:01:26.044",
        "framerate_reduction": 0,
        "h264_profile": "null",
        "id": "5678efgh",
        "keyframes": "follow_source",
        "name": "Video+Audio=Passthrough+Passthrough",
        "passthrough_audio": true,
        "passthrough_video": true,
        "stream_format": "audiovideo",
        "targets": [],
        "transcoder_id": "1234abcd",
        "updated_at": "2015-07-28T11:01:26.044"
    }
}
```

Create a stream target for the output
Next, configure a stream target to define the destination for the passthrough output rendition. You can use a custom stream target to deliver the stream to a third-party CDN or a use a Wowza CDN on Akamai stream target to deliver the stream to Wowza CDN.

**Add the stream target to the passthrough output**

Finally, add the custom stream target to the passthrough output rendition.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>The unique alphanumeric string that identifies the passthrough output rendition that will deliver content to the stream target. You can find the ID in the details of the passthrough output you just created.</td>
</tr>
<tr>
<td>stream_target_id</td>
<td>string</td>
<td>The unique alphanumeric string that identifies the stream target that will deliver the stream to viewers. You can find the ID in the details of the custom stream target you just created.</td>
</tr>
<tr>
<td>transcoder_id</td>
<td>string</td>
<td>The unique alphanumeric string that identifies the passthrough transcoder that will generate the output rendition. You can find the ID in the details of the transcoder you just created.</td>
</tr>
<tr>
<td>use_stream_target_backup_url</td>
<td>Boolean</td>
<td>Optional. Specify true to use the backup URL.</td>
</tr>
</tbody>
</table>
Example request and response

Add a target to an output:

```
curl -X POST \
-H "Content-Type: application/json" \
-H "wsc-api-key: ${WSC_API_KEY}" \
-H "wsc-access-key: ${WSC_ACCESS_KEY}" \
-d '{
  "output_stream_target": {
    "stream_target_id": "9123wxyz",
    "use_stream_target_backup_url": false
  }
}' "${WSC_HOST}/api/${WSC_VERSION}/transcoders/[transcoder_id]/outputs/[output_id]/output_stream_targets"
```

The details of the configured target are listed in the response, which should look something like this:

```
{
  "output_stream_target": {
    "stream_target_id": "9123wxyz",
    "use_stream_target_backup_url": false
  }
}
```

Related request

Remove a target from an output:

```
curl -X DELETE \
-H "wsc-api-key: ${WSC_API_KEY}" \
-H "wsc-access-key: ${WSC_ACCESS_KEY}" \
"${WSC_HOST}/api/${WSC_VERSION}/transcoders/[transcoder_id]/outputs/[output_id]/output_stream_targets/[output_stream_targets_id]"
```

Start and stop the transcoder

When the passthrough transcoder, output, and target are created, use the PUT
method to start and stop the transcoder. Wowza Streaming Cloud will pass the stream through the transcoder and send the output to the target address.

Example requests

Start the 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]/start"
```

Stop the 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]/stop"
```

Related requests

View a transcoder’s state:

```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]/state"
```

Possible transcoder states are starting, stopping, started, stopped, and resetting.

View the details of a running transcoder:

```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]/state"
```

Start all of a transcoder’s stream targets:
Stop all of a transcoder's stream targets:

```
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]/enable_all_stream_targets"
```

View a transcoder's preview image:

```
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]/thumbnail_url"
```

Delete a transcoder:

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