Learn how to use the REST API to ingest a single video encode into the Wowza Streaming Cloud™ service, transcode it to multiple adaptive bitrate renditions, and deliver the ABR output to a target, or destination. Then, learn how to programmatically start and stop the transcoder.

Create a transcoder

Start by creating a transcoder that receives the stream from a source encoder.

Transcoder parameters

<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>
</tbody>
</table>

Notes:
- **eu_ireland** doesn't support passthrough streaming or 24x7 billing.
- **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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buffer_size</td>
<td>integer</td>
<td>The size, in milliseconds, of the incoming buffer, which stores packets before they’re processed to enable efficient transcoding. Valid values are 0, 1000, 2000, 3000, 4000, 5000, 6000, 7000, and 8000. The default is 4000.</td>
</tr>
<tr>
<td>delivery_method</td>
<td>string</td>
<td>The method you’re using to deliver the source stream to the transcoder. Use one of the following valid values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>pull</strong> (Default) Instructs Wowza Streaming Cloud to pull the stream from the RTMP or RTSP source.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>push</strong> - Instructs the RTMP or RTSP source to push the stream directly to Wowza Streaming Cloud.</td>
</tr>
<tr>
<td>low_latency</td>
<td>Boolean</td>
<td>Speeds the time it takes to decode and deliver video data to the player by turning off the sort packet buffer. The default is false. Specify true.</td>
</tr>
</tbody>
</table>
The name of the transcoder. Enter an alphanumeric string that is short (maximum 200 characters) and descriptive, for example, MyABRtranscoder.

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

Required for RTMP and RTSP pull connections. Enter the source encoder’s web address, without the preceding protocol or trailing slash (/).

Optional for RTMP and RTSP push connections. Some encoders automatically append an extension to their stream names. If the device you’re using does this, enter the extension.

A dynamic buffer that helps stabilize streams in rough network conditions but adds latency. Recommended when pulling Akamai legacy RTMP streams or from a stream source. The default is false. Specify true to enable stream smoothing.

Specify the default, transcoded.

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

Example request and response
Create a transcoder:
The command creates a 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:

curl -X POST \
-H "Content-Type: application/json" \
-H "wsc-api-key: ${WSC_API_KEY}" \
-H "wsc-access-key: ${WSC_ACCESS_KEY}" \
-d '{
  "transcoder": {
    "billing_mode": "pay_as_you_go",
    "broadcast_location": "us_west_california",
    "buffer_size": 4000,
    "delivery_method": "push",
    "low_latency": true,
    "name": "MyABRtranscoder",
    "protocol": "rtsp",
    "transcoder_type": "transcoded"
  }
}' "${WSC_HOST}/api/${WSC_VERSION}/transcoders"
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 the highest bitrate output for the transcoder

Next, define the output renditions you want the transcoder to generate, starting with the highest bitrate rendition: a passthrough output 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>

```
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]"
```
Note: Additional parameters are available for outputs, but they don’t apply to passthrough output. For information on other output parameters, see the Wowza Streaming Cloud REST API Reference Documentation.

Example request and response

Create a passthrough output:

```bash
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:

```json
{
    "output": {
        "bitrate_audio": 0,
        "bitrate_video": 0,
        "created_at": "2015-07-28T11:26.044",
        "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": "2015-07-28T11:26.044"
    }
}
```
Create additional, lower bitrate outputs for the transcoder

Create as many additional outputs as you want the transcoder to generate. Additional outputs should be transcoded to create lower-quality renditions than the passthrough output. Depending on the resolution of the passthrough, you might want to create three to five additional outputs. For each additional output, specify the aspect ratio, bitrate, and profile you want to use.

Additional output rendition parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aspect_ratio_height</td>
<td>integer</td>
<td>The height, in pixels, of the output rendition. Should correspond to the aspect ratio (widescreen or standard) of the video source and be divisible by 8. The default is 1080.</td>
</tr>
<tr>
<td>aspect_ratio_width</td>
<td>integer</td>
<td>The width, in pixels, of the output rendition. Should correspond to the aspect ratio (widescreen or standard) of the video source and be divisible by 8. The default is 1920.</td>
</tr>
<tr>
<td>bitrate_audio</td>
<td>integer</td>
<td>The audio bitrate, in kilobits per second (Kbps). Must be between 0 (for passthrough) and 1000. The default is 128.</td>
</tr>
<tr>
<td>bitrate_video</td>
<td>integer</td>
<td>The video bitrate, in kilobits per second (Kbps). Must be between 0 (for passthrough) and 10240. The default is 4000.</td>
</tr>
<tr>
<td>framerate_reduction</td>
<td>string</td>
<td>Reduce the frame rate of the transcoded output rendition when streaming in 4K at high frame rates. The default, 0, uses the encoded stream's frame rate without reduction.</td>
</tr>
<tr>
<td>h264_profile</td>
<td>string</td>
<td>The encoding method. Specify main for desktop streaming, baseline for playback on mobile devices, or high for HD playback. The</td>
</tr>
</tbody>
</table>
default is **high**.

<table>
<thead>
<tr>
<th>keyframes</th>
<th>string</th>
<th>Specify <strong>follow_source</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>passthrough_audio</strong></td>
<td>Boolean</td>
<td>Specify <strong>false</strong>.</td>
</tr>
<tr>
<td><strong>passthrough_video</strong></td>
<td>Boolean</td>
<td>Specify <strong>false</strong>.</td>
</tr>
</tbody>
</table>

**Example requests**

If your passthrough rendition is 1280x720, you might want to create four additional outputs.

Create an 848x480 output:

```bash
curl -X POST
-H "Content-Type: application/json" \ 
-H "wsc-api-key: ${WSC_API_KEY}" \ 
-H "wsc-access-key: ${WSC_ACCESS_KEY}" \ 
-d '{
    "output": {
        "aspect_ratio_height": 480,
        "aspect_ratio_width": 848,
        "bitrate_audio": 128,
        "bitrate_video": 1700,
        "framerate_reduction": 0,
        "h264_profile": "main",
        "keyframes": "follow_source",
        "passthrough_audio": false,
        "passthrough_video": false,
        "stream_format": "audiovideo"
    }
}' 
"${WSC_HOST}/api/${WSC_VERSION}/transcoders/[transcoder_id]/outputs"
```

Create a 640x360 output:
Create a 512x288 output:

```
curl -X POST \
-H "Content-Type: application/json" \
-H "wsc-api-key: ${WSC_API_KEY}" \
-H "wsc-access-key: ${WSC_ACCESS_KEY}" \
-d '{
  "output": {
    "aspect_ratio_height": 360, 
    "aspect_ratio_width": 640, 
    "bitrate_audio": 128, 
    "bitrate_video": 1024, 
    "framerate_reduction": 0, 
    "h264_profile": "main", 
    "keyframes": "follow_source", 
    "passthrough_audio": false, 
    "passthrough_video": false, 
    "stream_format": "audiovideo"
  }
}' "${WSC_HOST}/api/${WSC_VERSION}/transcoders/[transcoder_id]/outputs"
```

Create a 320x188 output:

```
curl -X POST \
-H "Content-Type: application/json" \
-H "wsc-api-key: ${WSC_API_KEY}" \
-H "wsc-access-key: ${WSC_ACCESS_KEY}" \
-d '{
  "output": {
    "aspect_ratio_height": 288, 
    "aspect_ratio_width": 512, 
    "bitrate_audio": 128, 
    "bitrate_video": 512, 
    "framerate_reduction": 0, 
    "h264_profile": "baseline", 
    "keyframes": "follow_source", 
    "passthrough_audio": false, 
    "passthrough_video": false, 
    "stream_format": "audiovideo"
  }
}' "${WSC_HOST}/api/${WSC_VERSION}/transcoders/[transcoder_id]/outputs"
```

Create a 320x188 output:
Create a stream target for the outputs

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

Add the stream target to each output
Finally, add the stream target to each 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 output rendition. You can find the ID in the details of each of the outputs 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 transcoder that will generate the output renditions. 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 <strong>true</strong> to use the target’s backup URL, if one is specified. The default is <strong>false</strong>.</td>
</tr>
</tbody>
</table>

**Example request and response**

Assign a stream target to an output:
The details of the assigned target are listed in the response, which should look something like this:

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

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

```json
{
   "output_stream_target": {
      "stream_target_id": "9123wxyz"
   }
}
```

Related request

Update a custom stream target configuration:

```bash
curl -X PATCH \
-H "Content-Type: application/json" \
-H "wsc-api-key: $(WSC_API_KEY)" \
-H "wsc-access-key: $(WSC_ACCESS_KEY)" \
-d '{
   "stream_target": {
      "name": "MyNewTargetName"
   }
}' "${WSC_HOST}/api/${WSC_VERSION}/stream_targets/custom/[custom_stream_target_id]"
```

Start and stop the transcoder

When the transcoder, outputs, and targets are created, use the PUT method to start and stop the transcoder. Wowza Streaming Cloud will ingest the stream from the source, create the passthrough output and the lower-quality transcoded output renditions, and send all of the outputs 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]/stats"
```

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]"
```