You can use the Wowza Streaming Cloud™ service REST API to restrict access to a stream by applying token authentication to a Wowza CDN on Akamai stream target. Token authentication allows only viewers who have the token, which is hashed and appended to the playback URL, to access the stream. You can use token authentication with Wowza CDN on Akamai stream targets to make the stream playback URL unavailable after a certain length of time, to limit access to approved IP addresses, or apply other restrictions. A common use is to protect pay-per-view content to only paying viewers.

**Note:** This article applies to Wowza CDN on Akamai stream targets only. To secure Wowza CDN on Fastly stream targets with token authentication, see [Protect a Wowza CDN on Fastly stream target with token authentication using the Wowza Streaming Cloud REST API](#).

### About token authentication

Token-based authentication uses a multipart token that consists of a delimited list of string fields. One field is an HMAC, or keyed-hash message authentication code. HMAC is a common mechanism for message authentication that uses cryptographic hash functions. The HMAC portion of the token hashes a trusted shared secret that you create in Wowza Streaming Cloud. It is short-lived and secures initial access to the stream.

The second part of the token, a cookie, is valid for the duration of a stream and protects segments that are delivered during playback. It restricts access to the stream according to query parameters that you specify. For example, you can expire the stream after a certain length of time or only allow whitelisted IP addresses to access it.

You append the token to the stream target’s playback URL, and then Wowza Streaming Cloud only lets viewers receive the content after it verifies the presence and validity of the token.

Token authentication is managed by the browser. No configuration is required for the player. However, token authentication requires that the viewer’s browser supports cookies.

**Notes:**
- Token authentication only works in Safari if the security preference `Accept Cookies` is set to `Always`. Otherwise, the protected stream can’t be played.
- Token authentication works with third-party players and with Wowza Player Builder. It doesn’t work with a player created in the Wowza Streaming Cloud live stream workflow and embedded in a hosted or third-party webpage.
- If using Wowza Player Builder, enable the `withCredentials` configuration property. See [Customize Wowza Player with configuration properties](#) for more information.

### Add a Wowza CDN on Akamai stream target for HLS playback

**Important:** Token authentication for Wowza CDN on Akamai stream targets is available in v1.3 of the Wowza Streaming Cloud REST API only. The feature is not available for Wowza CDN on Akamai targets in v1.4 and later.

First, add a Wowza CDN on Akamai stream target configured to play streams from Wowza CDN over HLS or HLS and HDS.

### Stream target parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data</th>
<th>Description</th>
</tr>
</thead>
</table>

---

**Note:** This article applies to Wowza CDN on Akamai stream targets only. To secure Wowza CDN on Fastly stream targets with token authentication, see [Protect a Wowza CDN on Fastly stream target with token authentication using the Wowza Streaming Cloud REST API](#).
<table>
<thead>
<tr>
<th><strong>Type</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>A descriptive name for the stream target. Maximum 255 characters.</td>
</tr>
<tr>
<td>provider</td>
<td>Specify akamai_cupertino (for HLS-only playback) or akamai (for HLS and HDS playback).</td>
</tr>
<tr>
<td>use_cors</td>
<td>(Optional) CORS, or cross-origin resource sharing, allows streams to be safely delivered across domains. For example, they can be sent to providers such as Peer5, Viblast, and Streamroot, which implement a peer-to-peer grid delivery system. Specify true to enable CORS. The default is false. Available if the provider is akamai_cupertino.</td>
</tr>
<tr>
<td>use_secure_ingest</td>
<td>(Optional) Specify true to deliver the HLS stream securely between the transcoder and the target. The default is false. Available if the provider is akamai_cupertino.</td>
</tr>
</tbody>
</table>

**Example request and response**

The following request generates a Wowza CDN on Akamai stream target for HLS playback:

```bash
curl -X POST \
-H "Content-Type: application/json" \
-H "wsc-api-key: ${WSC_API_KEY}" \
-H "wsc-access-key: ${WSC_ACCESS_KEY}" \
-d '{
  "stream_target_wowza": {
    "name": "MyHLSTarget",
    "provider": "akamai_cupertino"
  }
}' "${WSC_HOST}/api/${WSC_VERSION}/stream_targets/wowza"
```

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

```
{
  "stream_target_wowza": {
    "created_at": "2016-02-23T16:04:23.170Z",
    "hls_playback_url": "https://[wowzasubdomain]-i.akamaihd.net/hls/live/[appname]/[streamname]/playlist.m3u8",
    "id": "1234abcd",
    "name": "MyHLSTarget",
    "primary_url": "http://[wowzasubdomain]-i.akamaihd.net/[appname]/[streamname]",
    "provider": "akamai_cupertino",
    "stream_name": "st1r2eam",
    "updated_at": "2016-02-23T16:04:23.170Z",
    "use_cors": false,
    "use_secure_ingest": false
  }
}
```

**Create token authentication for the target**

Create token authentication for the Wowza CDN on Akamai stream target.

**Token authentication parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enabled</td>
<td>Boolean</td>
<td>Specify true to permit token authentication for the target.</td>
</tr>
<tr>
<td>trusted_shared_secret</td>
<td>string</td>
<td>Provide a trusted shared secret for the token authentication. Must contain only hexadecimal characters and be an even number of characters equal to or less than 32.</td>
</tr>
</tbody>
</table>

**Example request and response**
The following request enables token authentication for the target 1234abcd:

curl -X POST \
-H "Content-Type: application/json" \
-H "wsc-api-key: ${WSC_API_KEY}" \
-H "wsc-access-key: ${WSC_ACCESS_KEY}" \
-d '{
  "token_auth": {
    "enabled": true,
    "trusted_shared_secret": "12345678abcdefgh"
  }
}' \
"${WSC_HOST}/api/${WSC_VERSION}/stream_targets/1234abcd/token_auth"

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

```
{
  "token_auth": {
    "created_at": "2016-10-23T16:04:22.828Z",
    "enabled": true,
    "stream_target_id": "1234abcd",
    "trusted_shared_secret": "12345678abcdefgh",
    "updated_at": "2016-10-23T16:04:22.828Z"
  }
}
```

**Important:** After enabling token authentication for a Wowza CDN on Akamai stream target using REST API version v1.3, you must contact Support in order for the changes to take effect.

Related requests

View the details of a token authentication:

curl -X GET \
-H "wsc-api-key: ${WSC_API_KEY}" \
-H "wsc-access-key: ${WSC_ACCESS_KEY}" \
"${WSC_HOST}/api/${WSC_VERSION}/stream_targets/[stream_target_id]/token_auth"

Update the token authentication applied to a stream target:

curl -X PATCH \
-H "Content-Type: application/json" \
-H "wsc-api-key: ${WSC_API_KEY}" \
-H "wsc-access-key: ${WSC_ACCESS_KEY}" \
-d '{
  "token_auth": {
    "enabled": "false"
  }
}' \
"${WSC_HOST}/api/${WSC_VERSION}/stream_targets/[stream_target_id]/token_auth"

**Important:** After updating token authentication for a Wowza CDN on Akamai target using REST API version 1.3, you must contact Support in order for the changes to take effect.

Assign the stream target to a transcoder

Assign the token-authorized stream target to a transcoder's output rendition(s).

**Note:** For an adaptive bitrate transcoder, you must assign the target to all of the transcoder's output renditions.
Add stream target 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 that will deliver content to the stream target. You can find the ID in the details of the output’s transcoder.</td>
</tr>
<tr>
<td>stream_target_id</td>
<td>string</td>
<td>The unique alphanumeric string that identifies the stream target. You can find the ID in the details of the Wowza stream target you just created.</td>
</tr>
<tr>
<td>transcoder_id</td>
<td>string</td>
<td>The unique alphanumeric string that identifies the transcoder.</td>
</tr>
</tbody>
</table>

Example request and response

The following request adds the token-authorized stream target `1234abcd` to the output rendition whose ID is `5678efgh` for the transcoder `9012ijkl`.

```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_stream_target": {
    "stream_target_id": "1234abcd",
    "use_stream_target_backup_url": false
  }
}' ${WSC_HOST}/api/${WSC_VERSION}/transcoders/9012ijkl/outputs/5678efgh/output_stream_targets/
```

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

```json
{
  "output_stream_target": {
    "stream_target_id": "1234abcd",
    "use_stream_target_backup_url": false
  }
}
```

Generate the hashed token

Once enabled, your playback URLs need to include the `hdnts` query parameter for playback to work. You can write a query in C, Java, PHP, Ruby, or other language. [Download and experiment with these query code samples.](#) Various parameters can be used along with the `trusted_shared_secret`. The following table includes a list of the basic parameters.

**Note:** All parameters are of the type `string`.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>algo</td>
<td>The algorithm use to generate the token. Options are <code>sha1</code>, <code>sha256</code> (the default), or <code>md5</code>.</td>
</tr>
<tr>
<td>end_time</td>
<td>The time that protected access to the stream ends, in UTC seconds. For example, <code>1478995200</code> for 13 November 2016 00:00:00 GMT, 24 hours from <code>start_time</code>.</td>
</tr>
<tr>
<td>ip</td>
<td>The IP address to restrict this token to.</td>
</tr>
<tr>
<td>key</td>
<td>The <code>trusted_shared_secret</code> from Wowza Streaming Cloud required to generate the token. Using the example in this article, the value would be <code>12345678abcdefgh</code>.</td>
</tr>
<tr>
<td>start_time</td>
<td>The time that protected access to the stream begins, in UTC seconds. For example, <code>1478908800</code> for 12 November 2016 00:00:00 GMT. Use <code>now</code> for the current time.</td>
</tr>
<tr>
<td><strong>token_name</strong></td>
<td>Parameter name for the token. The default is <strong>hdnts</strong>.</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td><strong>token_type</strong></td>
<td>Specify <strong>2.0</strong>, <strong>2.0.2</strong>, <strong>PV</strong>, or <strong>Debug</strong>.</td>
</tr>
<tr>
<td><strong>url</strong></td>
<td>The Wowza Streaming Cloud playback URL, which is the <strong>hls_playback_url</strong> of the Wowza Streaming Cloud stream target. Using the example in this article, the value would be <strong>https://[wowzasubdomain]-i.akamaihd.net/hls/live/[appname]/[streamname]/playlist.m3u8</strong>.</td>
</tr>
</tbody>
</table>

Attach the hashed token to the URL of your stream using the format

```
[playback_URL]?[token_name]=parameters_and_hashed_token
```

For example, for an HLS stream:

```
http://[wowzasubdomain]-i.akamaihd.net/hls/live/[appname]/[streamname]/playlist.m3u8?
hdnts=exp=1461972009~acl=/*~hmac=de43455a65009cbb538495e5bc70c9565a3c559406c0c7bc2a1cfeaff9344706
```

For an HDS stream:

```
http://[wowzasubdomain].akamaihd.net/z/[streamname_angle]@[stream_id]/manifest.f4m?
hdnts=exp=1461972009~acl=/*~hmac=de43455a65009cbb538495e5bc70c9565a3c559406c0c7bc2a1cfeaff9344706
```