Add custom playlist headers to Apple HLS manifests in Wowza Streaming Engine

This article describes how to add custom Apple HLS M3U headers into manifest responses created by Wowza Streaming Engine™ media server software. You can do this by configuring a property in Wowza Streaming Engine Manager or programmatically by using the Wowza Streaming Engine Java API.

Note: Wowza Streaming Engine™ software version 4.4.0 or later is required.

About Apple HLS headers

Clients request Apple HLS (Cupertino) streams from Wowza Streaming Engine in two stages. The first stage is to request the playlist. A common request URL would be:

http://[wowza-ip-address]:1935/vod/mp4:sample.mp4/playlist.m3u8

Where [wowza-ip-address] is the server’s IP address. Wowza Streaming Engine responds to this request with output like this:

#EXTM3U
#EXT-X-VERSION:3
#EXT-X-STREAM-INF:BANDWIDTH=868638,CODECS="avc1.66.21,mp4a.40.2",RESOLUTION=512x288
chunklist_w345834349.m3u8

In the second stage, clients request the chunklist provided in the initial server response. The chunklist request is relative to the playlist, for example:

http://[wowza-ip-address]:1935/vod/mp4:sample.mp4/chunklist_w345834349.m3u8

Wowza Streaming Engine responds to this request with output like this:

EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:10
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:10.0, media_w345834349_0.ts
#EXTINF:10.0, media_w345834349_1.ts
#EXTINF:10.0, media_w345834349_2.ts

Both responses include EXT headers. You can include additional headers in the responses by configuring a property in Wowza Streaming Engine Manager or by using the Wowza Streaming Engine Java API. This article describes both methods.

Add Apple HLS headers using Wowza Streaming Engine Manager

To add Apple HLS manifest headers to Wowza Streaming Engine chunklist responses, you must add a property to your application configuration. The property you add depends on whether it’s a live or VOD the application and on whether the chunklist contains audio and/or video.

1. In Wowza Streaming Engine Manager, click the Applications tab and then click the name of your application.
2. On the application page Properties tab, click Custom in the Quick Links bar.

Note: Access to the Properties tab is limited to administrators with advanced permissions. For more information, see Manage credentials.

3. In the Custom section, click Edit, and then click Add Custom Property.
4. In the Add Custom Property dialog box, specify the Path, Name, Type, and Value for any of the properties in the following two tables, depending on the type of application and chunklist.

Cupertino manifest header properties for VOD applications
### Cupertino manifest header properties for live applications

<table>
<thead>
<tr>
<th>Path</th>
<th>Name</th>
<th>Type</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>/Root/Application/LiveStreamPacketizer</td>
<td>cupertinoManifestHeadersAudio</td>
<td>String</td>
<td>EXT-CUSTOM-TEST:nameone=valueone</td>
<td>EXT-CUSTOM-TEST:nametwo=2</td>
</tr>
</tbody>
</table>

**Note:** For the value, to add multiple parameters to the same header or to include multiple headers, you can delimit header names, parameters, and values with a pipe character (|).

---

5. Click **Add**, click **Save**, and then restart the application.
With a property added, the Wowza Streaming Engine chunklist response to the client includes the new headers, for example:

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:10
#EXT-X-MEDIA-SEQUENCE:0
#AUTHENTICATE:authname=authvalue
#EXT-CUSTOM-TEST:namethree,nametwo=2,nameone=valueone
#EXT-CUSTOM-TEST2
#EXTINF:10.0, media_w345834439_0.ts
#EXTINF:10.0, media_w345834439_1.ts
#EXTINF:10.0, media_w345834439_2.ts
```

In this example response, two `EXT-CUSTOM-TEST` parameters are merged into one header and the `AUTHENTICATE` header is included separately.

### Add Apple HLS headers using the Wowza Streaming Engine Java API

Use the Wowza Streaming Engine Java API to programmatically add Apple HLS manifest headers to chunklist responses. This section provides examples that show how to use the API for on-demand streaming, live streaming, and live streaming with digital rights management.

#### On-demand streaming example

For on-demand streams, the API is available via the HTTP session for the specified stream. The following example code shows that the `onHTTPCupertinoStreamingSessionCreate` event is called when a client connects.

```java
public void onHTTPCupertinoStreamingSessionCreate(HTTPStreamerSessionCupertino httpSession)
{
    if (httpSession instanceof HTTPStreamerSessionCupertino)
    {
        HTTPStreamerSessionCupertino httpSessionCupertino = (HTTPStreamerSessionCupertino)httpSession;
        CupertinoUserManifestHeaders userManifestHeaders = httpSessionCupertino.getUserManifestHeaders();
        if (userManifestHeaders != null)
        {
            userManifestHeaders.addHeader("EXT-CUSTOM-TEST", "nameone", "valueone");
            userManifestHeaders.addHeader("EXT-CUSTOM-TEST", "nametwo", 2);
            userManifestHeaders.addHeader("EXT-CUSTOM-TEST", "namethree");
            userManifestHeaders.addHeader("EXT-CUSTOM-TEST2");
        }
    }
}
```

Although any headers can be added, only certain headers are included in the manifest response, depending on what other media server functionality is enabled. For example, the `EXT-X-KEY` header is only included in the manifest if digital rights management (DRM) is enabled, so even if you add a custom parameter for the `EXT-X-KEY` header, it won’t be included in the manifest response unless DRM is enabled for the session.

#### Live streaming example

For live streams, the API is available via the live stream packetizer for the specified stream. The following example code shows that the `IHTTPStreamerCupertinoLivePacketizerDataHandler2` interface is called when a client connects.
Live streaming with DRM example

For live streams with DRM, the API is available via the live stream packetizer for the specified stream. The following example code shows that additional methods are called each time an Apple HLS chunk is created to give you the opportunity to control how that chunk is encrypted.
The above example code shows that additional **EXT-X-KEY** headers are mapped. Note that the URI element will replace the URI that’s normally sent back. This provides an opportunity to replace the value with an attribute that works with custom clients and software.

The `{encKeySessionId}` and `{query}` variables in the URI are translated to a session ID and query parameters (if appropriate) before they’re included in the manifest response to the client. This can be useful in some implementations to allow per-session key requests on any DRM server.

**More resources**

- Secure Apple HLS streaming using DRM encryption with Wowza Streaming Engine