EZDRM Configuration
Apple FairPlay License Delivery

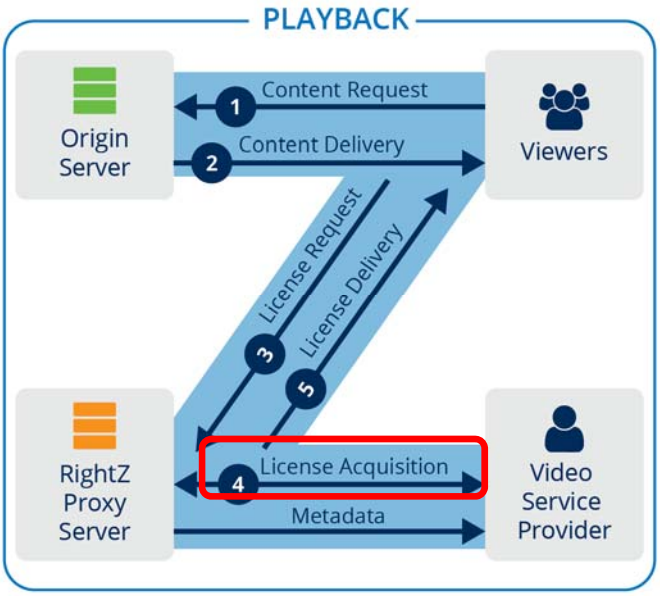
Table of Contents

Version 1.0

Apple FairPlay DRM Setup Overview

EZDRM Apple FairPlay DRM is a hosted Apple FairPlay Streaming (DRM). This enables a content owner to encrypt the media with Apple FPS DRM keys and deliver content Apple devices with native support MAC Safari browser via HTML 5 player or iOS via native App.

The EZDRM solution is composed of two separate processes. The first is encrypting your media, this is call the packaging process. This is accomplished via a secure web call to the EZDRM Key Servers API. The Key server API will return an XML response with the DRM key structure, these encryption key values are what will be used by any compliant 3rd party encoder and packager application. These are outlined in our packaging documents for Wowza, Bitmovin, MP4Box, and Bento 4 Open Source, etc.



The next part of the process is the issue of a DRM license for the media asset for playback. This document outlines the Playback and License acquisition process, as highlighted above in Playback step #4.

The next part of the process is the issue of a DRM license for the media asset. When the media player loads the asset, the FPS compliant player will read the HLS manifest and automatically call the EZDRM license servers. The EZDRM license servers will then open the request and match it to your account. Then the EZDRM servers will call

an Authentication URL that is hosted on your web infrastructure. This call will pass the metadata of the player along with any additional custom data such as session details, user identification or any custom details that is needed for your business logic to run. Upon receiving this request, you can process any business logic, and either return the DRM rules you wish EZDRM to issue or deny the request.

Playback License Delivery

Licenses are issued via an authentication URL call-back method, where the player will send a DRM license request to EZDRM servers. EZDRM will then make a back-end web call to the client Authorization URL. This URL is hosted on your web infrastructure and you will use this to process they business logic to grant licenses via returning license values.

Persistent vs. Non-Persistent Licenses

There are two types of licenses issued; persistent and non-persistent.

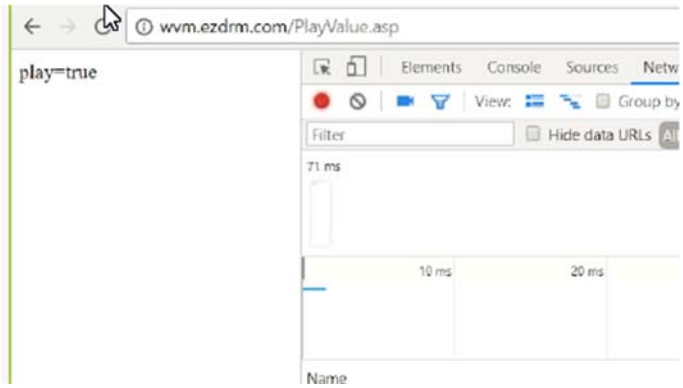
Persistent Licenses – license data is stored locally on the device and has a persistent location where the license stored. Repeated accessing of the media will not result in a new license call. If the license becomes expired or invalid, it will go through the licensing process again and store the new license data. Examples of browsers that support persistent licenses include Microsoft Internet Explorer (IE) and Edge. Android devices as well as apps such as Netflix and Google Play also support persistent licenses because they offer a persistence license locker for storage. Changing devices or settings can result in a new license call and a new playback license.

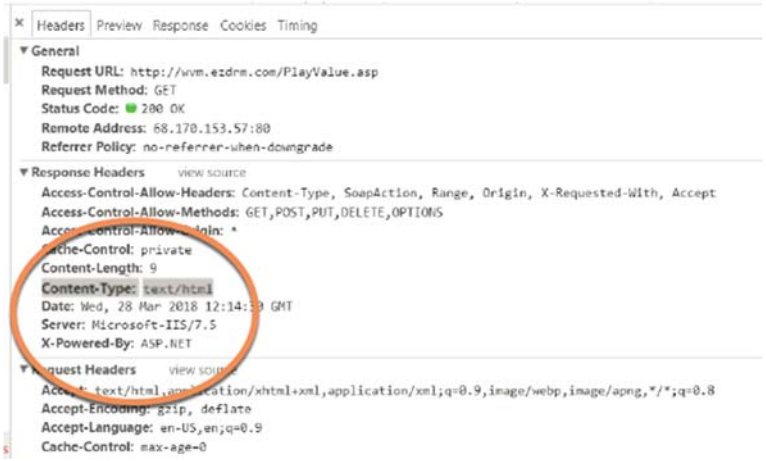
Non-Persistent Licenses – many browsers do not have a persistent place to store license data. Each time the media is accessed, a new license call will be made resulting in a new non-persistent playback license. Examples of browsers that do not support persistent licenses are Firefox, Chrome and HTML5 players. If you are using a browser, EZDRM will assume that you are requesting a non-persistent license because not all browsers are supported. For instance, Widevine does not currently support Chrome or Firefox, although that could change in the future.

Apple FairPlay License Delivery

This generic authentication URL is a simple example of the FairPlay response: http://fps.ezdrm.com/demo/play.asp

FairPlay responses need be in text format (not in html format). The Authorization URL can be a dynamic URL or a flat TXT file, but both return back values in TXT format. Be sure that the response **Content-Type** is set to **text/html** format.





Simple and Detailed responses

Simple response

The simple response **play = true** assumes the following values:

```
Offline=False
```

```
Play=True
Rental_Duration=0
```

When using a browser, or other media that does not support persistent licenses, you can send the response **play=true** and EZDRM assumes the default settings above. There would be no need to return any additional values.

The descriptions of these values are as follows:

Parameter	Description	Value Type
Offline	Allows a persistent or non-persistent license; True=persistent license; False=no persistent license.	True/False
Play	Gives the media permission to play	True/False
Rental_Duration	Time window in seconds that playback is permitted, a value of 0 indicates that there is no limit to the duration	Seconds; default is 0

Detailed response

When using an app, or both a browser and an app you can send a more detailed license response, including requesting a persistent license. If a detailed response is returned, EZDRM will return back those explicit attributes.

If you issue a persistent license to a device or browser that does not support persistent licenses, it will automatically be ignored and treated like a non-persistent license.

Sending a detailed response will allow you to set each specific attribute like the example below:

```
Offline=True
Play=True
```

```
Rental_Duration=6000
```

This response is for a license that lasts for 10 Minutes and will be stored locally on the client's device.

Passing Custom Data

As part of the licensing process, the content owner can pass a set of parameters through the EZDRM system in order to use that information within the business logic. These parameters should be attached to the server URL provided above.

This is an example in the HTML player:

```
var serverProcessSPCPath  = '<<FPS PATH TO LICENSE SERVER>>/api/licenses/<<ASSET ID>>?<<KEY1>>=<<VALUE1>>&<<KEY2>>=<<VALUE2>>';
var serverProcessSPCPath  = 'http://fps.ezdrm.com/api/licenses/09cc0377-6dd4-40cb-b09d-XXXXXXXX?CustomValue=Value1?CustomValue2=Value2';
```

The EZDRM system passes several values by default to your Authorization URL:

- The FPS License Server Path
- AssetID – changes per the asset; the AssetID is generated during DRM key generation.
- Custom Data values
- The Client IP of the end client is added to the end of the string

Here is an example of the EZDRM return post to your Authorization URL:

http://fps.ezdrm.com/api/licenses/09cc0377-6dd4-40cb-b09d-XXXXXXXX?customdata=123?customdata=345&ClientIP=12.34.567.89

Sample Player

For an iOS offline example, you can download using this link:
http://www.ezdrm.com/downloads/fps-ios-demo-updates-3.zip

To use the player for testing you'll need to:

- Edit the Cert file
- Edit the m3u8