# PallyCon License Token Guide V1.0

## Overview {#intro}

There are two types of methods for issuing multi-DRM (FPS, Widevine, PlayReady, NCG) licenses from PallyCon cloud server.

1. Callback type

    - When PallyCon cloud server receives license request from multi-DRM client, it first checks service site's callback page to see if the user has valid permissions.
    - In the case of a request from an authorized user, the service site returns information such as authentication, usage rights (unlimited, fixed period) and various security options to the PallyCon cloud server through the callback web page.
    - PallyCon cloud server receives the response from the callback page and issues the license to the client.

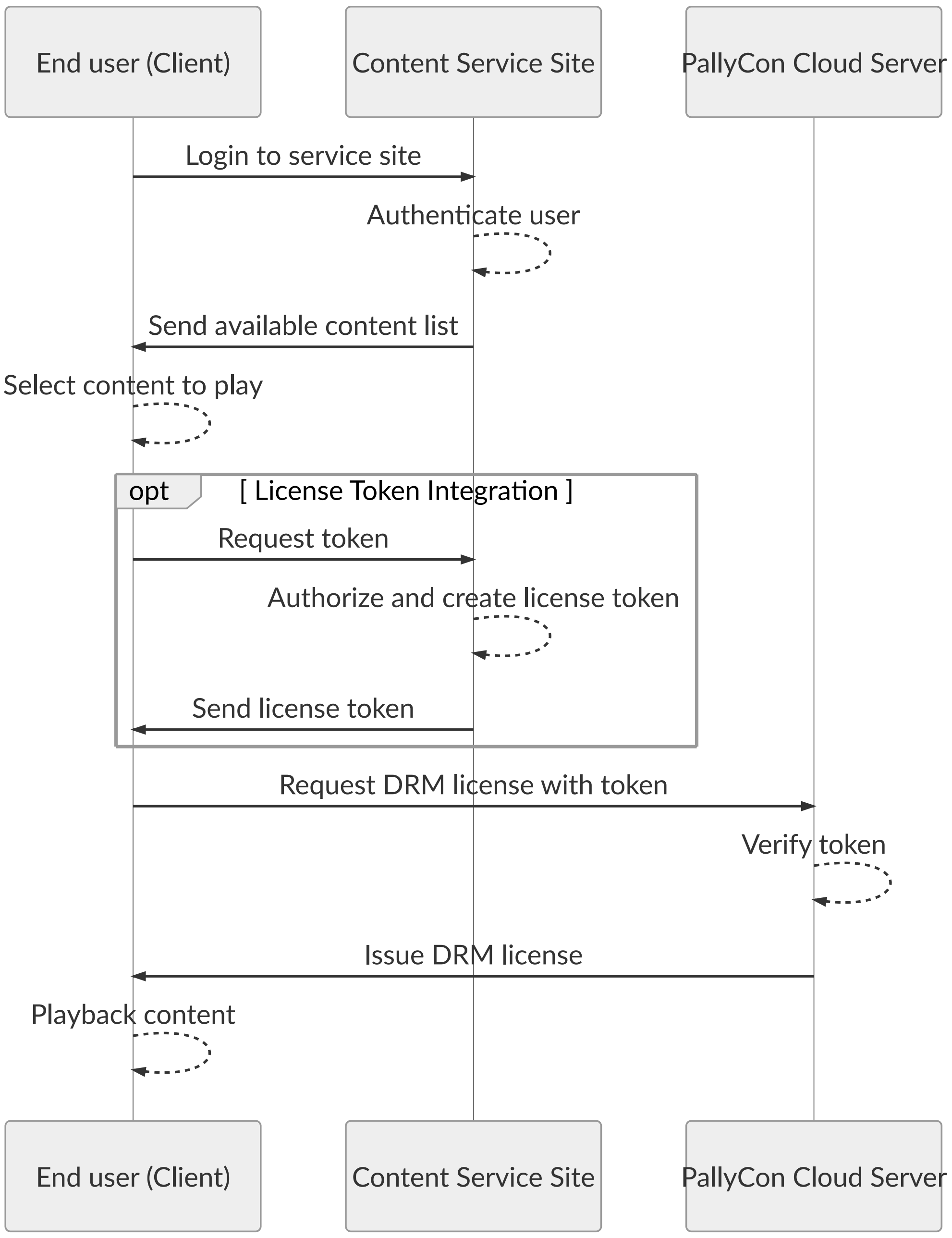
2. Token type

    - When a multi-DRM client tries to play DRM content, the client requests a token to the service site in order to acquire DRM license. The service site verifies that the user requesting the token has permission to the content, and then generates a token data according to the specification.
    - The service site can set usage rights (expiration date or unlimited) and various security options inside of the token data. The generated token is delivered to the client as response.
    - When a client requests a license with a token, the PallyCon cloud server validates the token and issues a license.

This document describes the second method, the specification of license token. Please refer to License Callback Guide if you want callback type integration.

> Step-by-step instructions and sample code for how to generate tokens can be found in the License Token Tutorial documentation.

## Token License Issuance Flow {#workflow}



(1) Request a token to service site

- Client requests its service site for a token to playback DRM content.

(2) Token generation (see Specification below)

- The service site checks the request received from the client and generates a token if the user has permission to use the content.
- Token includes information such as Content ID, user ID, timestamp, and license rules.

(3) Token transfer

- The service site returns the generated token to the client as a response.

> The creation of the token and the request / response function are not provided by PallyCon. The service site should implement these functions according to this guide.

(4) Request for a license

- The client places the received token(base64 string) in pallycon-customdata-v2 and requests license to PallyCon cloud server.

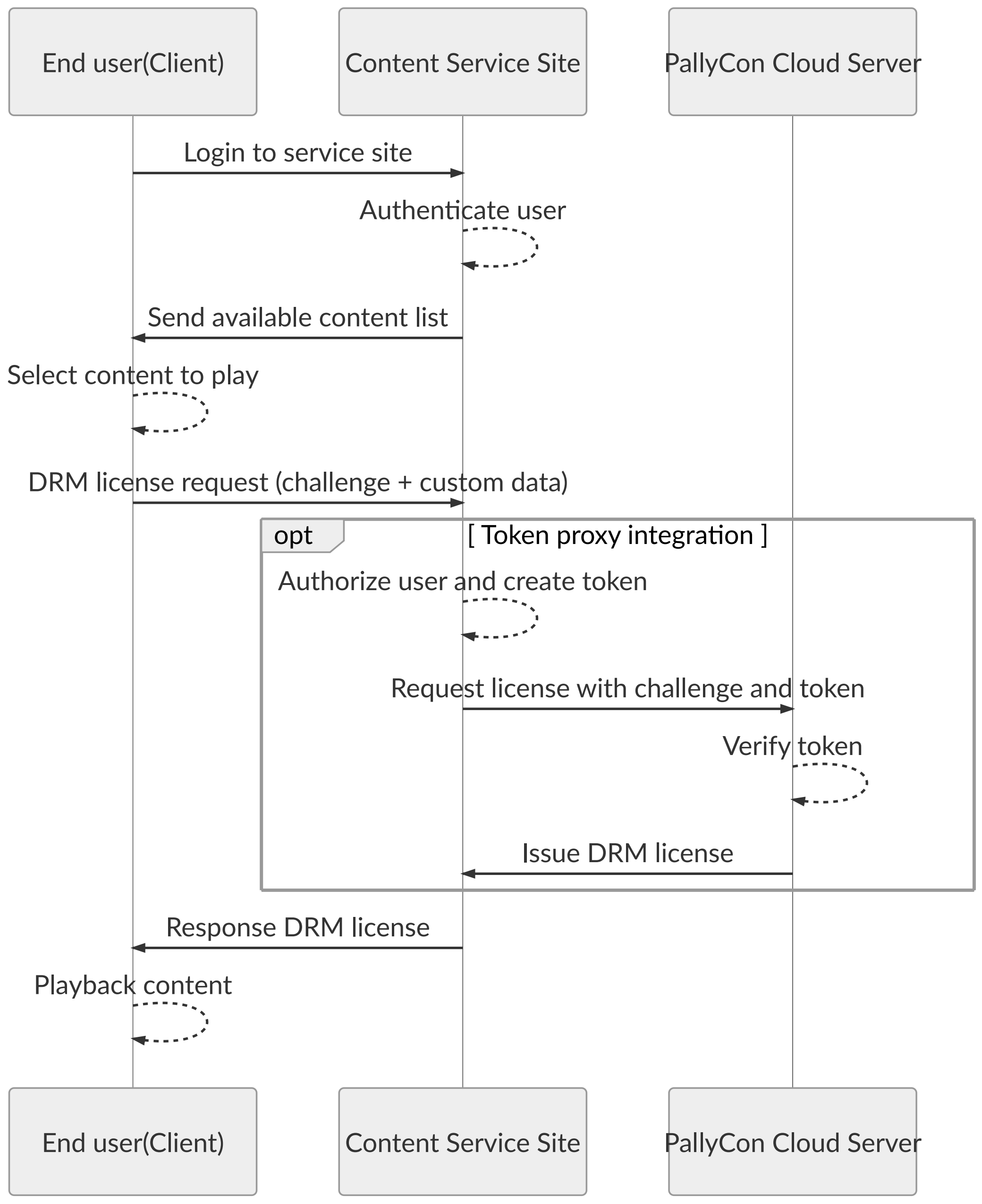
> For licensing requests through CustomData, please refer to the Multi-DRM License Integration Guide or Client SDK Guide.

(5) Issuing a license

- PallyCon cloud server validates the token and issues a license according to the rules in the token.

## Token Proxy Type License Issuance Flow

Token-based license issuance can also be handled through a proxy server at the service site, as shown below.



(1) Request DRM license to proxy server

- A client requests a DRM license to a proxy server of the service site for DRM content playback.
- The client calls the URL of the proxy server instead of PallyCon license server URL by DRM LA_URL configuration. The User ID and Content ID may be sent via custom header or URL parameter of the license request.
- License request data sent to the proxy server include challenge data generated by the client's DRM module.

(2) Authorize the user and create token

- Service site checks if the user has a right for the content using the custom data sent to the proxy server.

- The proxy server generates a license token with DRM license rules for the service's business model and security policy.

(3) Request DRM license to PallyCon server

- The proxy server requests a DRM license to PallyCon license server with the generated token and the license challenge data sent from the client.
- PallyCon server validates the token and issues DRM license using the challenge data.

(4) Response license and playback content

- The proxy server delivers the DRM license data issued by the PallyCon license server to the client.
- The client player starts playing content using the delivered DRM license.

## Token Generation Specification {#token-json}

- The service site generates the following JSON token for the request from the client and sends the base64 encoded string as the response.

- You can test generating token by entering key values of the following specification in DevConsole page of PallyCon site.

### Token JSON Format

```
{
    "drm_type": "<drm type string>",
    "site_id": "<site id string>",
    "user_id": "<user id string>",
    "cid": "<content id string>",
    "policy": "<base64(aes256(token policy for license generation))>",
    "timestamp": "<token validity start time (GMT) as yyyy-mm-ddThh:mm:ssZ>
    "hash": "<base64(sha256(hash message format))>"
}
```

| Name | Value | Required | Description |
|---|---|---|---|

| Name | Value | Required | Description |
|---|---|---|---|
| drm_type | string | No | Type of DRM ("NCG", "Widevine", "PlayReady", "FairPlay"), case sensitive, Default : "PlayReady" |
| site_id | string | Yes | Service Site ID which is issued by PallyCon Console |
| user_id | string | Yes | End-user's ID which is managed by the service site. Input "LICENSETOKEN" if there is no user ID. |
| cid | string | Yes | Unique ID of the content. The CID is used in DRM content packaging as well. (Max 200 bytes alphanumeric string) |
| policy | base64 encoded string | Yes | Token Policy in JSON (refer to specification) which is encrypted by AES256 and encoded as Base64 string. |
| timestamp | string | Yes | Token validity start time (usually current time) as 'yyyy-mm-ddThh:mm:ssZ'(GMT). Token is valid for 600 seconds after the timestamp. (can be adjusted on Console site) |
| hash | base64 encoded string | Yes | Hash message (refer to specification) which is hashed by SHA256 and encoded as Base64 string. |

> Value of `drmType` should be entered exactly in case according to the specification.

## Token JSON example

```
{
    "drm_type":"Widevine",
    "site_id":"ABCD",
    "user_id":"LICENSETOKEN",
    "cid":"sample-content-id-0123",
    "policy":"uZ0ALHJDHdZKc9pICii6Hog46frSIl+to/Wbf08uqliQVjGwK0Lw40onRM743
    "timestamp":"2018-04-14T23:59:59Z",
    "hash":"QkM4NDVGMDMxRUE4MDM0NUMzQUE4MTgyMTA4QTQ2QjQyNEFBNTJCNkQ1QjhGODg
}
```

> Note:
>
> The token and hash strings shown in the above example are not valid data. They are for reference only. For practical application, use the values generated according to this specification.

### Example of Token String

- The below string is the result of Base64 encoding for the above JSON example.

```
ewogICAg4oCcZHJtX3R5cGXigJ064oCdV2lkZXZpbmXigJ0sCiAgICDigJxzaXRlX2lk4oCdOuK
```

## Token Policy JSON Format {#token-policy-json}

- Encrypt the JSON value configured in the following format with AES256, and set the Base64 encoded string as the ‘policy’ value of the Token JSON.
- For AES256 encryption method, refer to specifications.

```
{
    "playback_policy": {
        "limit": <true|false>,
        "persistent": <true|false>,
        "duration" : <int(seconds)>,
        "expire_date": "<license expiry time (GMT) as yyyy-mm-ddThh:mm:ssZ>
    },
    "security_policy": {
        "hardware_drm": <true|false>,
        "output_protect": {
            "allow_external_display" : <true|false>,
            "control_hdcp": <0|1|2>
        },
        "allow_mobile_abnormal_device" : <true|false>,
        "playready_security_level": <150|2000>
    },
    "external_key": {
        "mpeg_cenc": {
            "key_id" : "<hex-string>",
            "key" : "<hex-string>",
            "iv" : "<hex-string>"
        },
        "hls_aes" : {
            "key" : "<hex-string>",
            "iv" : "<hex-string>"
        },
        "ncg":{
            "cek":"<hex-string>"
        }
    }
}
```

| Name | Value | Required | Description |
|---|---|---|---|
| playback_policy | json | No | license rules related with playback (refer to spec) |
| security_policy | json | No | license rules related with security (refer to spec) |
| external_key | json | No | Uses external content key to generate license. (refer to spec) |

### playback_policy {#playback-policy}

| Name | Value | Required | Description |
|---|---|---|---|
| limit | boolean | No | whether playback period is limited (default: false)<br>true : limited playback period, false : unlimited |
| persistent | boolean | No | whether the license is persistent. (default: false)<br>true : keep license, false : remove license after play(for streaming) |
| duration | number | Select | duration of playback (unit: second).<br>**'expire_date' is ignored if 'duration' is set.**<br>'limit' should be true to apply this setting. |

| Name | Value | Required | Description |
|---|---|---|---|
| expire_date | string | Select | date of license expiration, GMT Time 'yyyy-mm-ddThh:mm:ssZ' 'limit' should be true to apply this setting. This setting cannot be used with 'duration'. |

## security_policy {#security-policy}

| Name | Value | Required | Description |
|---|---|---|---|
| hardware_drm | boolean | No | Whether hardware DRM is required.(default: false) valid for CENC (Widevine Modular) contents only |
| output_protect | json | No | settings for external display (refer to spec) |
| allow_mobile_abnormal_device | boolean | No | whether rooted device is allowed (default: false) |
| playready_security_level | number | No | Security level of PlayReady DRM, 150,2000 (default: 150) |

### security_policy.output_protect {#output-protect}

| Name | Value | Required | Description |
|---|---|---|---|
| allow_external_display | boolean | No | Whether external display is allowed. (default: false) valid for NCG DRM only |
| control_hdcp | number | No | Setting for applying HDCP. (default: 0) 0 : No HDCP, 1 : HDCP 1.4, 2 : HDCP 2.2 |

## external_key {#external-key}

| Name | Value | Required | Description |
|---|---|---|---|
| mpeg_cenc | json | No | CENC external key setting for PlayReady/Widevine (refer to spec) |
| hls_aes | json | No | HLS AES external key setting for FairPlay Streaming (refer to spec) |
| ncg | json | No | NCG DRM external key setting (refer to spec) |

#### external_key.mpeg_cenc {#external-key-cenc}

| Name | Value | Required | Description |
|---|---|---|---|
| key_id | hex-string | No | Key ID for DASH CENC packaging(PlayReady/Widevine). 16byte hex string |
| key | hex-string | No | Key for DASH CENC packaging. 16byte hex string |
| iv | hex-string | No | IV for DASH CENC packaging. 16byte hex string |

#### external_key.hls_aes {#external-key-aes}

| Name | Value | Required | Description |
|---|---|---|---|
| key | hex-string | No | Key for HLS Sample AES packaging(FairPlay Streaming). 16byte hex string |
| iv | hex-string | No | IV for HLS Sample AES packaging. 16byte hex string |

#### external_key.ncg {#external-key-ncg}

| Name | Value | Required | Description |
|---|---|---|---|
| cek | hex-string | No | CEK for NCG packaging. 32byte hex string |

## Token Policy JSON Example

For basic integration test, refer to the below simple token policy. It will generate streaming license with 5 minutes of playback time limit.

```
{
    "playback_policy": {
        "limit": true,
        "persistent": false,
        "duration" : 300
    }
}
```

The below sample has more rules to show the full specification.

```
{
    "playback_policy":{
        "limit":true,
        "persistent":true,
        "duration":3600,
        "expire_date":"2018-04-20T23:59:59Z"
    },
    "security_policy":{
        "hardware_drm":true,
        "output_protect": {
            "allow_external_display":false,
            "control_hdcp":1
        },
        "allow_mobile_abnormal_device":false,
        "playready_security_level":150
    },
    "external_key": {
        "mpeg_cenc": {
            "key_id" : "30313233343536373839616263646566",
            "key" : "30313233343536373839616263646566",
            "iv" : "30313233343536373839616263646566"
        },
        "hls_aes" : {
            "key" : "30313233343536373839616263646566",
            "iv" : "30313233343536373839616263646566"
        },
        "ncg":{
            "cek":"30313233343536373839616263646566303132333435363738396162
        }
    }
}
```

## SHA256 Hash Message Format {#hash-message}

- The hash message is used to verify the integrity of the entire token JSON value and should be generated as follows:

```
base64( sha256( <site access key> + <drm type> + <site id> + <user id> + <c
```

1. Generate a string by concatenating the values of the access key of the service site and the values excluding the 'hash' field of the token JSON in order. Access key can be found on PallyCon Console site.
2. Generate the final hash message string by base64 encoding the sha256 hash value of the string created above.

> The resulting value of the sha256 hash function must be entered into the base64 function as a byte array type, not as a string.

### SHA256 Hash Message Example

```
Step 1. origin string
<Access Key>WidevineABCDLICENSETOKENsample-centent-id-0123uZ0ALHJDHdZKc9pIC

Step 2. sha256 + base64 string (final result)
QkM4NDVGMDMxRUE4MDM0NUMzQUE4MTgyMTA4QTQ2QjQyNEFBNTJCNkQ1QjhGODg1NUE1MDI2NjQ
```

## AES256 Encryption {#aes256}

- AES256 Encryption/Decryption should be processed as below using site authentication key which is created by 'Service Request' on PallyCon Console site. ( The key can be found on PallyCon Console's settings page )
- You can test AES256 encryption / decryption from the DevConsole page on the PallyCon site.

```
AES256 Encryption
- mode : CBC
- key : 32 byte (Site key from PallyCon Console site)
- iv : 16 byte (0123456789abcdef)
- padding : pkcs7
```