Widevine DRM
Proxy Integration

version 2.20

Contents

Revision History

Version	Date	Description
1.6	12/5/2014	- Expanded available parameters. - Clarification on string value types. - Added Appendix (License duration, Security levels, HDCP output)
1.7	1/29/2015	Additional edits after review.
1.9	10/12/2015	- Clarification on required / optional fields. - key id has max length of 16 bytes. - Expanded license response details including message_type, request_type, cert_serial_number.
2.0	4/15/2016	- Content_id is now optional in the request. - Added missing content_key_specs.iv field.
2.1	5/11/2016	Rental duration parameter has been deprecated, to be reinstated at a later date.
2.2	6/1/2016	Added section on how to protect external content keys. (draft)
2.3	6/9/2016	Completed section on how to protect external content keys.
2.4	6/29/2016	Added always_include_client_id.
2.5	7/7/2016	Clarification on key_id and key value generation.
2.6	12/19/2016	Updated renewal document links.
2.7	2/9/2017	- Corrected CGMS flags syntax. - Updated HDCP flags parameters. - Updated message_type parameters. - Updated allowed_track_type parameters. - Updated track_type parameters.
2.8	4/28/2017	- Updated license workflow with service certificate request. - Added service certificate and VMP sections.
2.9	5/9/2017	Added Test environment service certificate.
2.10	6/22/2017	Added client_id_msg.
2.11	7/24/17	- Added disable_analog_output to required_output_protection. - Added sd_only_for_l3 to license request.
2.12	8/30/17	Introduce the Proxy SDK component.
2.13	9/14/17	Added srm_version and srm_included to license metadata response.
2.14	1/28/2018	Updated license durations (time windows) description, operation and behavior.

2.15	5/3/2018	● Updated VMP status codes.
2.16	6/6/2018	● Added HDCP table for Chromecast.
2.17	7/27/2018	● Added offline license release. ● Updated HDCP tables for Android.
2.18	3/6/2019	● Added allow_unverified_platform.
2.19	10/23/2019	● Updated document and service certificate links.
2.20	1/10/2020	● Updated VMP information.

Introduction

This document describes how external content providers integrate their license proxy component with the Widevine Cloud License Service. This solution is intended for content providers who have their own proxy servers for user authentication and access authorization, but relies on Widevine service for policy and content key management and generating the license.

System Diagram
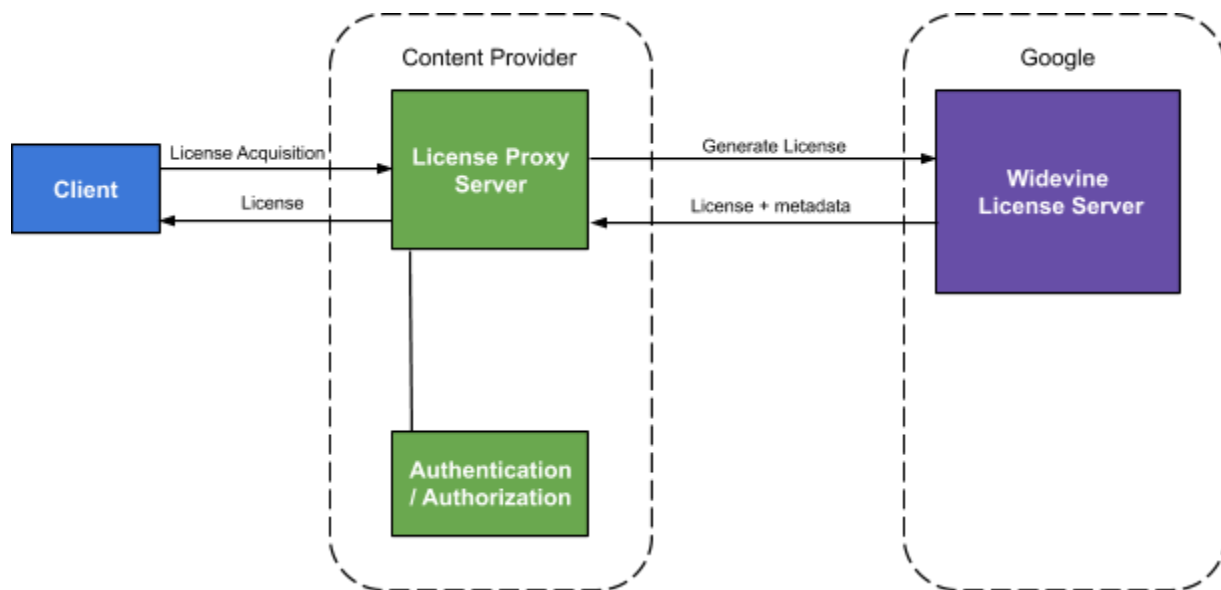
All communication between client, proxy and license service is via HTTPS.

Workflow

The certificate request workflow is **required** for Chrome 59 and later, iOS and specific use-cases in ChromeOS.

The PARSE_ONLY workflow is **optional**, used to inspect the incoming encrypted request payload, increases latency. Recommend using Proxy SDK instead.

Proxy SDK provides APIs to manage the service certificate request directly and inspect the Widevine license request..

1. A client **may** generate a service certificate request.
    a. The proxy must sign the certificate request.
2. A client generates a license request and sends it to the license proxy.
    a. The proxy authenticates the user and authorizes the request.
3. Providers not using the Proxy SDK:
    a. This **may** include a requirement to view the license request details via a PARSE_ONLY request.
4. Proxy SDK is used to query the license for information such as device make and model.

5. The proxy determines the desired business rules (if any).
6. The proxy calls GetLicense API using HTTP POST.
7. Widevine License service receives the license request
   a. obtains content keys
   b. generates a license
   c. returns license to the proxy.
8. The proxy inspects GetLicense response status and forwards the license payload blob (as byte-stream) to client.

GetLicense API

This API allows a Content Provider proxy to fetch a Modular DRM license from the Widevine Cloud License Service.

Request

The request is an HTTP POST request to the license URL. The license URL is provided with your service credentials. Please contact us for more details.

The POST body is JSON formatted data in the following syntax:

```
{
        "request": "<message>",
        "signature": "<signature>"
        "signer" : <signer name>"
}
```

Name	Value	Description
request	Base64 encoded string.	The actual message. This is a serialized JSON message containing the request or the response. The message is in the clear.
signature	Base64 encoded string.	AES CBC encryption of the SHA1 hash of <message>

Credentials are provided by Widevine. Please contact us for more details. |
| signer | String | Identifies the entity sending the message.

Credentials are provided by Widevine. Please contact us for more details. |

<message> is a JSON formatted string containing the license challenge, policy overrides and license configurations. The syntax is

```
        {
                "payload": "<license challenge>",
                "content_id": "<content id>"
                "provider":"<provider>"
                "allowed_track_types": "<types>",
                "content_key_specs": [
                        { "track_type": "<track type 1>" },
                        { "track_type": "<track type 2>" }, … ],
                "policy_overrides": {
                        "can_play": <can play>,
                        "can persist": <can persist>,
                        "can_renew": <can renew>,
                        "rental_duration_seconds": <rental duration>,
                        "playback_duration_seconds": <playback duration>,
                        "license_duration_seconds": <license duration>,
                        "renewal_recovery_duration_seconds": <renewal recovery
duration>,
                        "renewal_server_url":"<renewal server url>",
                        "renewal_delay_seconds":<renewal delay>,
                        "renewal_retry_interval_seconds":<renewal retry interval>,
                        "renew_with_usage":<renew with usage>
                }
        }
```

Fields below are optional unless marked as required.

Name	*Value*	*Description*
payload	Base64 encoded String	**REQUIRED** The license request sent by a client.
provider	String	**REQUIRED** Used to look up content keys and policies. Credentials are provided by Widevine. Please contact us for more details.
content_id	Base64 encoded string max 36 bytes	This is the content identifier. Used to associate KeyId(s) and Content Key(s) for each content_key_specs.track_type. Note: convert your string to base64.
~~policy_name~~ (**deprecated**)	~~String~~	~~Name of a previously registered policy.~~ ~~This field is provided for backwards compatibility with legacy Widevine systems. As a general~~

		~~recommendation, all business rules should be specified in the license request according to this document.~~ ~~Do not use this parameter unless you are a Widevine Classic legacy user.~~

allowed_track_types	Enum, one of:
	`SD_ONLY`
	`SD_HD`
	`SD_UHD1`
	`SD_UHD2`
		Controls which content keys should be included in a license.
		
		SD_ONLY - returns SD and AUDIO keys only
		SD_HD - returns SD, HD and AUDIO keys only
		SD_UHD1 - returns SD, HD, UHD1 and AUDIO keys
		SD_UHD2 - returns SD, HD, UHD1, UHD2 and AUDIO keys (all)

content_key_specs	Array of JSON structures (see [Content Key Spec](#))	A finer grained control on what content keys to return. See [Content Key Spec](#) below for details. Only one of allowed_track_types and content_key_specs can be specified.
use_policy_overrides_exclusively	Boolean. `true or false`	Use policy attributes specified by policy_overrides and omit all previously stored policy.
policy_overrides	JSON structure (see [Policy Overrides](#))	Policies settings for this license. In the event this asset has a predefined policy, these specified values will be used.
session_init	JSON structure (see [Session Initialization](#))	Optional data passed to license.
parse_only	Boolean. `true or false`	The license request is parsed but no license is issued. However, values from the license request are returned in the response.
session_key	Base64 encoded string.	If specified, the content [keys](#) and [IVs](#) specified in [ContentKeySpec](#) are encrypted with this key using AES-CBC (PKCS5 padding). The session_key itself is encrypted with the partner's AES key that was provided by Widevine.
session_iv	Base64 encoded string.	**REQUIRED** *if session_key was specified.* If specified, the content [keys](#) and [IVs](#) specified in [ContentKeySpec](#) are encrypted with this IV [using AES-CBC (PKCS5 padding)](#). The session_iv itself is encrypted with the partner's AES key that was provided by Widevine.

client_id_msg	String of binary bytes	Results from using the Proxy SDK to call *GetClientInfoAsString()*. The Proxy SDK is a set of API designed to allow easier integration with Widevine DRM licensing. This field would be populated by partners who manage their own service certificate.
sd_only_for_l3	Boolean. `true or false`	Only return AUDIO and SD keys if this device is Widevine L3. This takes precedence over "allowed_track_types". *Default = false.*
allow_unverified_platform	Boolean `true or false`	A license request will fail if VMP status is unverified or tampered for a desktop browser. Set this field to **'true'** to allow license request to succeed when VMP status is **unverified**. *Default = false*.

`license_start_time` is the time when the license is being requested. Playback is halted when `license_start_time` + `license_duration_seconds` is exceeded.

Policy Overrides

Fields below are optional unless marked as required.

Name	*Value*	*Description*
policy_overrides. can_play	Boolean. `true or false`	**REQUIRED** Indicates that playback of the content is allowed. *Default = false.*
policy_overrides. can_persist	Boolean. `true or false`	Indicates that the license may be persisted to non-volatile storage for offline use. *Default = false*.
policy_overrides. can_renew	Boolean `true or false`	Indicates that renewal of this license is allowed. If true, the duration of the license can be extended by heartbeat. *Default = false.*
policy_overrides. license_duration_seconds	Int64	Indicates the time window for this specific license. A value of 0 indicates unlimited. *Default = 0.* See [Appendix]().
policy_overrides. rental_duration_seconds	Int64	Indicates the time window while playback is permitted. A value of 0 indicates unlimited. *Default = 0.* See [Appendix]().
policy_overrides. playback_duration_seconds	Int64	The viewing window of time once playback starts within the license duration. A value of 0 indicates unlimited. *Default = 0.* See [Appendix]().
policy_overrides. time_shift_limit_seconds	Int64	Indicates the allowed delay between the time the content was transmitted and the time the content is viewed. A value of 0 indicates unlimited. *Default = 0.*

		See Appendix.
policy_overrides. renewal_server_url	String	All heartbeat (renewal) requests for this license shall be directed to the specified URL. This field is only used if can_renew is true.
policy_overrides. renewal_delay_seconds	int64	How many seconds after license_start_time, before renewal is first attempted. This field is only used if can_renew is true. *Default = 0.*
policy_overrides. renewal_retry_interval_seconds	int64	Specifies the delay in seconds between subsequent license renewal requests, in case of failure. This field is only used if can_renew is true. *Default = 0.*
policy_overrides. renewal_recovery_duration _seconds	Int64	The window of time, in which playback is allowed to continue while renewal is attempted, yet unsuccessful due to backend problems with the license server. A value of 0 indicates unlimited. This field is only used if can_renew is true. *Default = 0.*
policy_overrides. renew_with_usage	Boolean `true or false`	Indicates that the license shall be sent for renewal when usage is started. This field is only used if can_renew is true. *Default = false.*
policy_overrides. always_include_client_id	Boolean `true or false`	Indicates to clients that license renewal and release requests must include client identification (client_id). *Default = false.*

Session Initialization

This structure is used to pass optional data when issuing a license.

Fields below are optional unless marked as required.

Name	*Value*	*Description*
provider_client_token	Base64 encoded string	Client token to send back in the license response. If the license request contains a client token, this value is ignored. The client token will persist beyond license sessions. *provider_client_token* is only supported in the Chrome CDM and *persistentState* must be enabled by the Application.
override_provider_client_token	Boolean. `true or false`	If false and the license request contains a client token, use the token from the request even if a client token was specified in this structure. If true, always use the token specified in this structure. *Default = false.*
session_id	Base64 encoded string	Identify the session. This value will exist in all **subsequent license renewals** associated with this license.

Content Key Specs

If a pre-existing policy exists, there is no need to specify any of the values in the Content Key Spec. The pre-existing policy associated with this content will be used to determine the output protection such as HDCP and CGMS. If a pre-existing policy is not registered with the Widevine License Server, the content provider can inject the values into the license request (recommended approach).

Each content_key_specs must be specified for all tracks, regardless of the option *use_policy_overrides_exclusively*.

Fields below are optional unless marked as required.

Name	*Value*	*Description*
content_key_specs.track_type	String Options: SD HD AUDIO UHD1 UHD2	A track type definition. If content_key_specs is declared in the license request, you must specify **all track types explicitly** for playback. AUDIO - audio tracks SD - 576p or less HD - 720p, 1080p UHD1 - 4K UHD2 - 8K
content_key_specs.security_level	uint32	Defines client robustness requirements for playback. 1 - Software-based whitebox crypto is required. (SW_SECURE_CRYPTO) 2 - Software crypto and an obfuscated decoder is required. (SW_SECURE_DECODE) 3 - The key material and crypto operations must be performed within a hardware backed trusted execution environment. (HW_SECURE_CRYPTO) 4 - The crypto and decoding of content must be performed within a hardware backed trusted execution environment. (HW_SECURE_DECODE) 5 - The crypto, decoding and all handling of the media (compressed and uncompressed) must be handled within a hardware backed trusted execution environment. (HW_SECURE_ALL) *Default = 1.*

		See [Appendix](#) for more information.

content_key_specs.required_output_protection.hdcp	String - one of:
	`HDCP_NONE,`
	`HDCP_V1,`
	`HDCP_V2,`
	`HDCP_V2_1,`
	`HDCP_V2_2,`
	`HDCP_NO_DIGITAL_OUTPUT`
		Indicates whether HDCP is required.
		
		Default = HDCP_NONE.
		
		Note: These values must be capitalized.
		
		See [Appendix](#).

content_key_specs.required_output_protection.cgms_flags	String - one of:
	`CGMS_NONE,`
	`COPY_FREE,`
	`COPY_ONCE,`
	`COPY_NEVER`
		Indicates whether CGMS is required.
		
		Default = CGMS_NONE.
		
		Note: Do not specify this parameter for desktop browser platforms.
		Note: These values must be capitalized.

content_key_specs.required_output_protection.disable_analog_output	Boolean. `true or false`	Indicates whether analog output is allowed. *Default = false.*

content_key_specs.required_output_protection.hdcp_srm_rule	String - one of:
	`HDCP_SRM_RULE_NONE,`
	`CURRENT_SRM`
		Use *CURRENT_SRM* to not allow this key if the device has an older SRM and cannot support SRM updates.

content_key_specs.key	Base64 encoded string	Content key to use for this track. If specified, the track_type **OR** key_id is required. This option allows the content provider to inject the content key for this track instead of letting Widevine license server generate or lookup a key. **Note: convert your key value to binary and base64.**
content_key_specs.iv	Base64 encoded string	IV associated with the content key for this track.
content_key_specs.key_id	Base64 encoded string binary, 16 bytes	Unique identifier for the key. **Note: convert your string or hex value to binary and base64.**

Example of specifying and [protecting](#) content_key_specs

<table>
<tr><td align="center">***ContentKeySpecs (request)***</td></tr>
<tr><td>

```
    [
        {
            "key_id":"MGRmMjVjYzUtOWFkMS01MjhiLTkwZGMtNTYyY2QxODExODNi",
            "track_type":"SD",
            "key":"Pv38THb6rTQbIpzg04/qag==",
            "iv":"REVBREJFRUZERUFFEQkVFRg=="
        },
        {
            "key_id":"NzVlYThiYWMtZjg3Yy01YzMxLWFmYTYtNWY3YmUwNjE1MGU5",
            "track_type":"HD",
            "key":"xiE14IkMIRKxSPBQL+HksA==",
            "iv":"REVBREJFRUZERUFFEQkVFRg=="
        },
        {
            "key_id":"NDczMWNhNzItOWExYy01YjQ1LThjMTMtMGY4MDJiMDRiMWNj",
            "track_type":"AUDIO",
            "key":"y9DDm30ttT07GqXfxKMwdQ==",
            "iv":"REVBREJFRUZERUFFEQkVFRg=="
        }
    ]
```

</td></tr>
</table>

License Response

The response is in JSON format.

```
{
      "operation_status_code": <status code>,
      "licenses": <license>,
      "license_medata": [
      {
            "drm_status_code": <drm status code>,
            "asset_id":<asset identifier>,
            "policy": {
                      "organization_name":"<organization>",
                      "name":"<policy name>",
                      "track_policy": [
                      {
                      "track_type":"<track type>",
                      "level":<security level>,
                      "output_protection":"<output protection>"
```

```
                    }
                … ],
            }
        }
    ...]
}
```

Name	Value	Description
status	Uint32	Result of the request. *OK = 0;* *SIGNATURE_FAILED = 1;* *INVALID_LICENSE_CHALLENGE = 2;* *INVALID_CONTENT_INFO = 3;* *POLICY_UNKNOWN = 4;* *MALFORMED_REQUEST = 5;* *INTERNAL_ERROR = 6;* *PROVIDER_MISSING = 7;* *INVALID_REQUEST = 8;* *ACCESS_DENIED = 9;* *SIGNING_KEY_EXPIRED = 10;*
license	Base64 encoded string	License response that needs to be passed to the CDM after Base64 decode.
license_metadata	JSON structure (see License_Metadata)	License attributes.
make	String	Manufacturer of the device making the license request.
model	String	Model of the device making the license request.
security_level	Uint32 (see Client Security Level)	Widevine-defined security level as referenced within the EME specification.
drm_cert_serial_number	String	Globally unique serial number of certificate associated with this device. Example: `7f6f3150b373262cbdc3b2dade075dbb`
message_type	String - one of: `LICENSE_REQUEST`, `LICENSE`, `SERVICE_CERTIFICATE_REQUEST`, `SERVICE_CERTIFICATE`, `ERROR_RESPONSE`	Indicates the type of message contained in the license going back to the CDM. This field may be missing or empty if the server failed to generate a license or certificate.

License_Metadata

Name	*Value*	*Description*
content_id	String	content id used to generate the license.
license_type	Uint32	1 - indicates streaming. 2- indicates offline.
request_type	Uint32	1 - indicates NEW license. 2 - indicates RENEWAL license. 3 - indicates RELEASE license.
platform_verification_status	Uint32	See Appendix for values.
srm_version	Uint32	SRM version sent back to the client. This field only exist if the requesting policy sets hdcp_srm_rule = *CURRENT_SRM*.
srm_included	Boolean. `true or false`	Indicates if SRM in included in the license. This field only exist if the requesting policy sets hdcp_srm_rule = *CURRENT_SRM*.

Using Service Certificates

Service certificates are required by the Widevine client in the following circumstances:

- Chrome 59 (or later)
- Widevine iOS client
- ChromeOS (Chromebooks) - when remote_attestation is enabled.

It is highly recommended that the service certificate workflow is supported in all proxy implementations. The sample proxy scripts provide code examples on how to support this request.

The certificate request is a 2-byte binary payload which must be fulfilled by the license service. The certificate request must be signed.

Sample request and response certificate payload

```
2017-04-12 17:42:53,085 - proxy - DEBUG - Proxied Request = {"signer":
"widevine_test", "request": "eyJwYXlsb2FkIjogIkNBUT0ifQ==", "signature":
"1TQ71YXPYBsCI/A4tpmzoMOyNqzWboBf3VKvqIQ14is="}
2017-04-12 17:42:53,102 - proxy - DEBUG - response = {
  "status":"OK",
"license":"CAUSxQUKvwIIAxIQKHA0VMAI9jYYredEPbbEyBiL5/mQBSKOAjCCAQoCggEBALUhErjQ
XQI/zF2V4sJRwcZJtBd82NK+7zVbsGdD3mYePSq8MYK3mUbVX9wI3+lUB4FemmJ0syKix/XgZ7tfCsB
6idRa6pSyUW8HW2bvgR0NJuG5priU8rmFeWKqFxxPZmMNPkxgJxiJf14e+baq9a1Nuip+FBdt8TSh0x
hbWiGKwFpMQfCB7/+Ao6BAxQsJu8dA7tzY8U1nWpGYD5LKfdxkagatrVEB90oOSYzAHwBTK6wheFC9k
F6QkjZWt9/v70JIZ2fzPvYoPU9CVKtyWJOQvuVYCPHWaAgNRdiTwryi901goMDQoJk87wFgRwMzTDY4
E5SGvJ2vJP1noH+a2UMCAwEAAToSc3RhZ2luZy5nb29nbGUuY29tEoADmD4wNSZ19AunFfwkm9rl1Kx
ySaJmZSHkNlVzlSlyH/iA4KrvxeJ7yYDa6tq/P8OG0ISgLIJTeEjMdT/0l7ARp9qXeIoA4qprhM19cc
B6SOv2FgLMpaPzIDCnKVww2pFbkdwYubyVk7jei7UPDe3BKTi46eA5zd4Y+oLoG7AyYw/pVdhaVmzhV
DAL9tTBvRJpZjVrKH1lexjOY9Dv1F/FJp6X6rEctWPlVkOyb/SfEJwhAa/K81uDLyiPDZ1Flg4lnoX7
XSTb0s+Cdkxd2b9yfvvpyGH4aTIfat4YkF9Nkvmm2mU224R1hx0WjocLsjA89wxul4TJPS3oRa2CYr5
+DU4uSgdZzvgtEJ0lksckKfjAF0K64rPeytvDPD5fS69eFuy3Tq26/LfGcF96njtvOUA4P5xRFtICog
ySKe6WnCUZcYMDtQ0BMMM1LgawFNg4VA+KDCJ8ABHg9bOOTimO0sswHrRWSWX1XF15dXolCk65yEqz5
lOfa2/fVomeopkU",
  "supported_tracks":[],
  "internal_status":139,
  "message_type":"SERVICE_CERTIFICATE",
  "client_info":[]}
```

Pre-loading a service certificate

An alternative to executing the certificate request-response is to store the service certificate, prior to any license request. This method removes the overhead of an additional HTTPS round-trip to the license service.

The Widevine Cloud License Service certificate is only valid for use with the appropriate Widevine Cloud License Service:
- license.uat.widevine.com - [Download](#)
- license.widevine.com - [Download](#)

1. Download the tarball and extract the binary certificate file.
2. The client application must retrieve it (via HTTP GET) and execute EME's [setServiceCertificate API](#).
   - The certificate can be hosted on your service or proxy.
3. setServiceCertificate must be executed prior to every playback session to avoid the certificate request.

Verified Media Path (VMP)

The Verified Media Path (VMP) feature is implemented for desktop browser platforms. It describes our method to authenticate the browser software stack that is interfacing with the Widevine CDM.

VMP requires a valid service certificate.
VMP is available on the following platforms:
● Chrome 59 and later.

The Widevine license response will contain a platform_verification_status flag indicating if the browser framework is trusted.

Status	Description
PLATFORM_NO_VERIFICATION	The device platform does not support VMP. Example: Android, Tizen, WebOS, Roku
PLATFORM_UNVERIFIED	The device platform is browser-based (Chrome or ChromeOS) and cannot verify the hosting application (binary) of the Widevine client. For Windows or Mac OSX, the device is running a non-VMP-enabled application. On ChromeOS, the user boots the machine in a developer mode or blocks remote attestation (user permission prompt). All Linux systems return this status. By default, a license is not issued for this status code unless override is enabled.
PLATFORM_SOFTWARE_VERIFIED	The device platform status was verified at the software level. This is equivalent to Widevine device security 3.
PLATFORM_SECURE_STORAGE_SOFTWARE_VERIFIED	The device platform status was verified at the software level and the device has secure storage which is required for license storage persistence. This is equivalent to Widevine device security 3.
PLATFORM_HARDWARE_VERIFIED	The device platform status was verified at the hardware level. This is equivalent to Widevine device security 1.
PLATFORM_TAMPERED	The device platform status has been found to be tampered.

Sample license response metadata with VMP validation

This sample is from a Chromebook platform.

```
"License_metadata":{
"content_id":"bGxhbWFfd2lkZXZpbmVfdGVzdA==",
"license_type":"STREAMING",
"request_type":"NEW"},
"supported_tracks":[{"type":"SD",
"key_id":"kijTVq/KWk+aiSI9YVRlxQ=="},{"type":"AUDIO",
"key_id":"fQaj7GBJU06FEfRHVTGKYA=="},{"type":"HD",
"key_id":"7EDyLmsYXw6Uw4dtQSbELA=="}],
"make":"Google",
"model":"ChromeCDM-ChromeOS-x86",
"remote_attestation_verified":true,
...truncated...
"device_whitelist_state":"DEVICE_WHITELISTED",
"message_type":"LICENSE",
"platform":"",
"device_state":"RELEASED",
...truncated...
"platform_verification_status":"PLATFORM_HARDWARE_VERIFIED"}
```

Rejecting a request

A proxy may reject a license request if, for example, a user is not allowed to watch the video. In this case, a proxy may just return an empty HTTP response with appropriate response code, e.g. 403 Forbidden, or whatever response that the content provider's client application wishes to receive. No license blob is expected by the Widevine client module in this case.

Heartbeat (License Renewal) support

A proxy uses the same GetLicense API to obtain a heartbeat response. There is no need to pass in allowed_track_types and content_key_specs. A proxy controls whether the playback is allowed to continue by setting policy.can_play. The license renewals do not contain content keys, they only extend the duration of keys contained in the initial license.

- Related documentation
    - License Renewal Overview
    - Using License Renewal

Protecting proxy specified content keys

If the proxy is specifying content keys to the Widevine license service, it's strongly recommended the proxy encrypts the content keys. The process for the proxy is as follows:

1. Generate a random 16 byte session key and 16 byte session IV.
   - **Example session key:** `0x8727a40ebd82329e6b3b4e29fa3b004b`
   - **Example session IV:** `0x000102030405060708090a0b0c0d0e0f`
2. Encrypt each content key and content IV (AES-CBC, PKCS5 padding) using the session key and session IV. Example of using the following clear content key and iv.
   - **Clear content key:** `0x2038fa82910c404bee9200e8108baf29`
   - **Clear content IV:** `0x00112233445566778899aabbccddeeff`

   Encrypting the content key and iv with the session key/iv, results are:
   - Encrypted content key:
     `0x228107d15be6f2b0fc4945a3c7b2b0bac22a9bc0b154c1f19d181916ea008da5`
   - Encrypted content IV:
     `0x63a53152184e64f640992b7c6efa90c75890eaffd77200c0aadca51d9eab268a`
3. Encrypt the session key and session IV (AES-CBC, PKCS5 padding) using the provider's AES signing key/iv that is registered with Widevine. In this example, we'll use widevine_test signing key:
   - Widevine_Test (32 byte) signing key:
     `0x1ae8ccd0e7985cc0b6203a55855a1034afc252980e970ca90e5202689f947ab9`
   - Widevine_Test IV:
     `0xd58ce954203b7c9a9a9d467f59839249`

   Encrypting the session key and IV with the signing key, results are:
   - Encrypted session key:
     `0x2895244b8f656616c2813de6ebe6d3f55b803d5ce2f51ff1f78c2d4365f2aa1f`
   - Encrypted session IV:
     `0x1c1f219112972db8adb58e15d4ee95a7f9dbd813c308083798c275be939070ae`

4. Specify the Base64 encoded encrypted (binary) session key as the value to "`session_key`" in the JSON license request.
   - `session_key:KJUkS49lZhbCgT3m6+bT9VuAPVzi9R/x94wtQ2Xyqh8=`
5. Specify the Base64 encoded encrypted (binary) session IV as the value to "`session_iv`" in the JSON license request.
   - `session_iv:HB8hkRKXLbittY4V1O6Vp/nb2BPDCAg3mMJ1vpOQcK4=`

Status Codes

Code	Description
OK	Success.
SIGNATURE_FAILED	The server was unable to authenticate the message. Perhaps: ● The specified <signer> is unknown ● The <signature> was not specified, but is required in this case.
INVALID_LICENSE_CHALLENGE	The license challenge blob generated by client is invalid.
INVALID_WIDEVINE_HEADER	The Widevine header embedded in an encrypted asset is invalid.
POLICY_UNKNOWN	The specified <policy> was not found for this <signer>.
MALFORMED_REQUEST	The request is not formatted correctly.
INTERNAL_ERROR	Widevine server internal error.
INVALID_REQUEST	The request is not recognized, or is missing a required field.
CONTENT_ID_MISSING	Missing content_id
TRACK_TYPE_MISSING	Missing track_type
TRACK_TYPE_UNKNOWN	Unknown track_type
ACCESS_DENIED	
INVALID_WIDEVINE_PSSH	
TOO_MANY_CONTENT_SPECIFIERS	
CONTENT_ID_MISMATCH	Mismatched content_id
KEY_ID_MISMATCH	Mismatched key_id

Sample Proxy

A set of Python-based sample proxy scripts is available here. The package contains examples of signing the request, supporting a certificate request and injecting a custom content key.

Appendix

License Durations
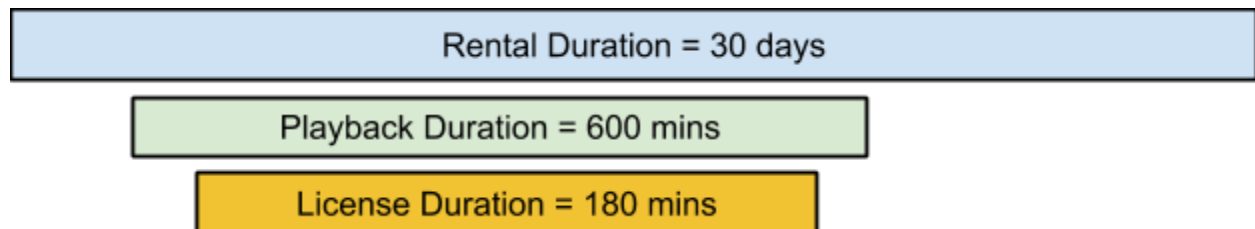
- License_Duration (license_duration_seconds) - Indicates the length of time to allow content playback.
    - This parameter is a hard value. Once this parameter is consumed (duration counts down to 0), no further decryption is permitted.
    - Minimum parameter required to limit content playback within a license.
- Playback_Duration (playback_duration_seconds) - Indicates the amount of time the license is valid after first (initial) use.
    - This parameter should only be specified in offline or renewal scenarios.
    - If playback duration is *unspecified*, it is unlimited.
- Rental_Duration (rental_duration_seconds) - Indicates the amount of time the license is valid before it is used.
    - Once use of the license has started, the Rental_Duration is not used or enforced.
    - This parameter should only be specified in offline scenarios.
    - If rental duration is *unspecified*, it is unlimited.

The additional duration parameters provide additional business logic:
- To enable support of license renewals.
    - With each successful renewal, the license duration is extended.
    - With continued successful renewals, the license duration can be extended to encompass the length of the media.
    - A renewal failure would result in premature license expiration.

The examples below illustrate the relationship and hierarchy of the duration parameters.

License for streaming content

A streaming use-case relies on an always-connected device:
- *Rental_Duration* is not specified.
- *Playback_Duration* is not specified.
- *License_Duration*
  - Set to 180 minutes as specified by proxy parameter.
- *License Expiration*
  - The license will activate (start expiring, counting down to 0) as soon as the license is received by the Widevine client on device.

License for streaming content with renewals
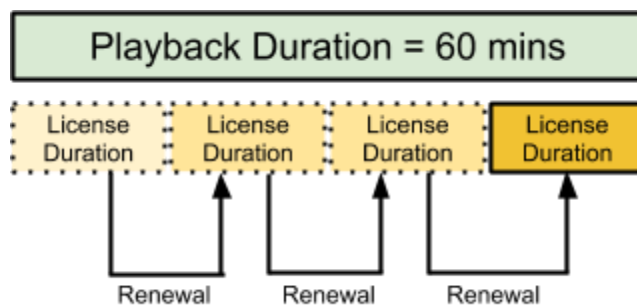
Like the above streaming example, *Rental_Duration* is not specified, but *Playback_Duration* is specified.
- *Playback_Duration*
  - Set to 60 minutes by proxy parameter.
- *Renewal_Delay*
  - Set to 15 minutes by proxy parameter.
- *License_Duration*
  - Set to 15 minutes internally by Widevine license server to match the *renewal_delay_interval*.
- *License Expiration*
  - The license will activate (start expiring, counting down to 0) as soon as the license is received by the Widevine client on device.

License for offline playback
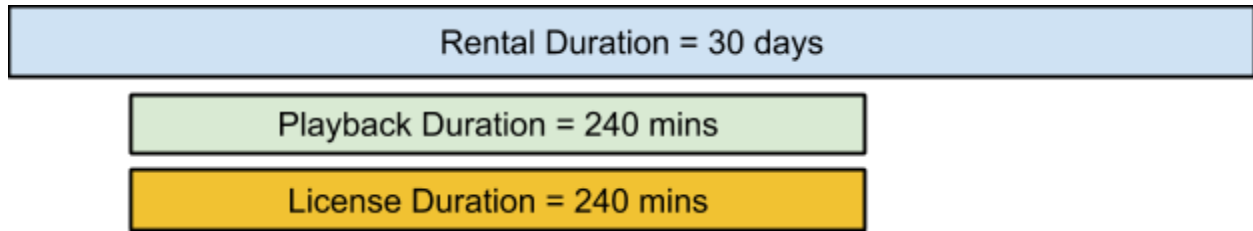
In the offline example, *Rental_Duration* and *Playback_Duration* are specified. In this example, the content is 120 minutes in length.

- *Rental_Duration*
    - Set to 43200 minutes (30 days) by proxy parameter.
- *Playback_Duration*
    - Set to 240 minutes by proxy parameter.
- *License_Duration*
    - Set to 240 minutes to allow user to complete viewing with ample time to spare (matches playback duration).
- *License Expiration*
    - The license will start expiring as soon as the license is first used for playback (User hits Play).

Client security level (content_key_specs.security_level)

A license may be issued with a requirement for the minimum client security level.

The table below illustrates the general mapping between the EME security level definitions and Widevine device robustness levels.

Definition	EME Level	Widevine Device Security Level
SW_SECURE_CRYPTO	1	3
SW_SECURE_DECODE	2	3
HW_SECURE_CRYPTO	3	2
HW_SECURE_DECODE	4	1
HW_SECURE_ALL	5	1

Note: For CrOS (Chromebooks), you will need to specify security_level=3 or higher to enable Widevine L1 support. Otherwise, the device will default to Widevine L3.

HDCP Enforcement

HDCP Parameters

HDCP Parameter	(applies to) WV Security Level	Description
HDCP_NONE	3	HDCP not specified
HDCP_V1	1	Enforce HDCP 1.x Playback not allowed if Client does not report HDCP 1.x or better.
HDCP_V2	1	Enforce HDCP 2.0 Playback not allowed if Client does not report HDCP 2.0 or better.
HDCP_V2_1	1	Enforce HDCP 2.1 Playback not allowed if Client does not report HDCP 2.1 or better.
HDCP_V2_2	1	Enforce HDCP 2.2 Playback not allowed if Client does not report HDCP 2.2 or better.
HDCP_NO_DIGITAL_OUTPUT	1	No digital output allowed, including HDCP. Internal display only.

Chrome

The table below is a results matrix for setting output protection HDCP_V1 in the license request. **Error** indicates playback will fail.

HDCP_V1	No External Display	HDCP External Display	Non-HDCP External Display	Analog External Display	ChromeCast	AirPlay
ChromeOS ARM	OK	OK	Error	N/A	Error	N/A
ChromeOS x86	OK	Error	Error	N/A	Error	N/A
Linux	Error	Error	Error	N/A	OK	N/A
Mac	OK	Error	Error	N/A	OK	Error
Windows	OK	OK	Error	N/A	OK	N/A

Chromecast

HDCP V1 Required	WV Security Level	Chromecast	Chromecast 2	NVidia Shield	Philips 32PFS6402/12
SW_SECURE_CRYPTO [1]	3	FAIL	OK	FAIL	FAIL
SW_SECURE_DECODE [2]	3	OK	OK	FAIL (OK*)	FAIL (OK*)
HW_SECURE_CRYPTO [3]	2	OK	OK	OK	OK
HW_SECURE_DECODE [4]	1	OK	OK	FAIL (OK*)	FAIL (OK*)
HW_SECURE_ALL [5]	1	OK	OK	FAIL (OK*)	FAIL (OK*)

Android

Widevine Security Level	HDCP version on Client (in license request)	HDCP version on Server (appended by proxy)	Playback Behavior
3	HDCP_NONE	HDCP version not specified	OK
3	HDCP_NONE	Any HDCP_V* parameter	Fail [1]
1	HDCP_NO_DIGITAL_OUTPUT	HDCP version not specified	OK [2]
1	HDCP_NO_DIGITAL_OUTPUT	Any HDCP_V* parameter	OK [2]
1	Any HDCP_V* parameter	Any HDCP_V* parameter if it is the same (or less than) as Client HDCP	OK

- Internal display only devices should report HDCP_NO_DIGITAL_OUTPUT.
- Level 3 devices with no HDCP capabilities should report HDCP_NONE.

[1] For Android L3, playback stops if output protection is required. L3 is best-effort and cannot be reliably enforced when specified.

[2] No output is allowed, therefore all output protection enforcement policies are met.

Sample Request (with signing)

Sending a licensing request to Widevine service:

wget -O /tmp/license --post-data
'{"request":
"ewogICJwYXlsb2FkIiA6ICJDQUVTaEFFS1RBZ0FFa2dBQUFBQ0FFQVWUFhhaHRRiL3E0M3YzU2Z1Qz
JWUDNxMGplQUVDVzNlbVFrV24yd1hDWVZPbnZsV1BETnFoOFZWSUI0R21zTkE4ZVZZWEdtlnWGtRVVlHT
jBHbGdNS2pwVUVUVEFvcUNNoUUlBUklRSk1QQ3psMmpXaXlNUUV0eGUsvZ3RtUkFCCR2hBeU5XWTNPRE16
TVRjeU1tSmpNMkV5R0FFZ3Y1aVFyQVVhSUMT04xelZnZVYywclA3dzJWbVZZMR29ycCNsY01RTzRCCRZGJ
IUHlyM0dzblkiLAogICJwcm92aWRlciIgOiAid2lkZXZpbmVfdGVzdCIsCiAgImNvbnRlbnRfaWQiOiOi
AiWm10cU0yeHFZVNk5rWm1Gc2EzSXphZz09IiwKICAiY29udGVudF9rZXlfc3BlY3MiOiBbCiAgICB7I
CJ0cmFja190eXBlIjogIlNEIiB9LAogICAgeyAidHJhY2tfdHlwZSI6ICJIRCIgfSwKICAgIHsgInRy
YWNrX3R5cGUiOiAiQVVESU8iIH0KICBdCn0K", "signature":
"xPkAbb3tjOY/ybdz0tmJMq9erH9ILnS5natMZr3QEW8=", "signer": "widevine_test" }'
https://license.uat.widevine.com/cenc/getlicense/widevine_test

The signature is generated as follows.
1. Take the raw value of the "request" field. Compute its SHA1 hash:
   xu5tN6lK+EEwMGA90tTbsP5upNY=
2. Encrypt the hash with AES CBC. The result is the signature. Convert it to base64 encoding and put it in the "signature" field.
   For integration tests, you can use "widevine_test" as the signer. Its AES key is
   ```
   0x1a, 0xe8, 0xcc, 0xd0, 0xe7, 0x98, 0x5c, 0xc0,
   0xb6, 0x20, 0x3a, 0x55, 0x85, 0x5a, 0x10, 0x34,
   0xaf, 0xc2, 0x52, 0x98, 0x0e, 0x97, 0x0c, 0xa9,
   0x0e, 0x52, 0x02, 0x68, 0x9f, 0x94, 0x7a, 0xb9
   ```
   The IV is
   ```
   0xd5, 0x8c, 0xe9, 0x54, 0x20, 0x3b, 0x7c, 0x9a,
   0x9a, 0x9d, 0x46, 0x7f, 0x59, 0x83, 0x92, 0x49
   ```

Expected Response

{"status":"OK","license":"CAIS6AIKKgoQMjVmNzgzMzE3MjJiY2NhMhIQMjVmNzgzMzE3MjJiY
zNhMhoAIAEoABIGCAEQARgBGGkYSELuvgFd3JF2KbY8Gr8wkSREaMNhP8g6Hn4Rdp7yG6F1/Fey5ZzNT
OJ0+G4IBaTQRg7L2HjGMK8CE6kjR4B9weSp7syABGkoKEAKUuVmddV3iu/D9yj+l6rcSEC46Gde9xaQ
N12q/Fgqv250aICR+Q+j0flfRYMdgbgC0jS+6RqFdXvvg94h5ZMKdZEU1IAIoARpKChDrZ2q7yzRelr
vPYWYw8aPaEhA2lTBnCeoVTeQ5bxydasrqGiDhgFnPjqr5W4S04cwzJaIlwMe0QJaakRZJ461DhIvvf
iACKAEaSgoQY52oDPI7VfO4yrP2TPpd9hIQGmUhfWKWnSgDGJjl9P/Lyhog6oKAeGNHF9AjeVJU47ri
qgKZh06qdMurDfKFywUMXUcgAigBIL+YkJAFOAAaIHP6Qy564dbSl+/ORuMlYm5U4CzSdtLBZkBn4W5
V+jTH","license_metadata":{"content_id":"ZmtqM2xqYVNkZmFFsa3Izag==","license_typ
e":"STREAMING","request_type":"NEW"},"supported_tracks":[{"type":"SD","key_id":
"ApS5WZ11XeK78P3KP6Xqtw=="},{"type":"HD","key_id":"62dqu8s0Xpa7z2FmMPGj2g=="},{
"type":"AUDIO","key_id":"Y52oDPI7VfO4yrP2TPpd9g=="}],"make":"","model":"","inte
rnal_status":0,"session_state":{"license_id":{"request_id":"MjVmNzgzMzE3MjJiYzN
hMg==","session_id":"MjVmNzgzMzE3MjJiYzNhMg==","purchase_id":"","type":"STREAMI

NG","version":0},"signing_key":"cEeLYQ+ZjldTOEnFy4e66+yhjo4jg6mMNiQqpEk0gBo=","keybox_system_id":4184,"license_counter":0},"cert_serial_number":"","device_whitelist_state":"DEVICE_WHITELISTED","message_type":"LICENSE","platform":"","device_state":"RELEASED","pssh_data":{"key_id":["JMPCzl2bViyMQEtyK/gtmQ=="],"content_id":""}}

Sample Request (injecting *clear* content keys)

This example includes content_key_specs. The JSON message before Base64 encoding is:

```
{
    "allowed_track_types":"SD_HD",
    "content_key_specs":[
        {
            "key_id":"MGRmMjVjYzUtOWFkMS01MjhiLTkwZGMtNTYyY2QxODExODNi",
            "track_type":"SD",
            "key":"Pv38THb6rTQbIpzg04/qag=="
        },
        {
            "key_id":"NzVlYThiYWMtZjg3Yy01YzMxLWFmYTYtNWY3YmUwNjE1MGU5",
            "track_type":"HD",
            "key":"xiE14IkMIRKxSPBQL+HksA=="
        },
        {
            "key_id":"NDczMWNhNzItOWExYy01YjQ1LThjMTMtMGY4MDJiMDRiMWNj",
            "track_type":"AUDIO",
            "key":"y9DDm30ttT07GqXfxKMwdQ=="
        }
    ],
    "use_policy_overrides_exclusively":true,
    "provider":"widevine_test",
    "content_id":"WU9VVFVCRTpDd3E3bWVXZWM3MA==",
    "policy_overrides":{
        "license_duration_seconds":36000,
        "can_play":true
    },
    "payload":"*<bytes-intentionally removed>*"
}
```

wget -O /tmp/license --post-data '{"signer": "widevine_test", "request":
"eyJhbGxvd2VkX3RyYWNrX3R5cGVzIjogIlNEX0hEIiwgImNvbnRlbnRfa2V5X3NwZWNzIjogW3sia2
V5X2lkIjogIk1HRm1NalZqWXpVtOWFkMS01MjhiLTkwZGMtNTYyY2QxODExODNi
CJ0cmFja190eXBlIjogIlNEIiwgImtleSI6ICJQdjM4VEhiNnJUUWJJcHpnMDQvcWFnPT0ifSwgeyJr
ZXlfaWQiOiAiTnpWbFlUaGlZV010Zjg3Yy01YzMxLWFmYTYtNWY3YmUwNjE1MGU5qRTFNR1U1Iiw
gInRyYWNrX3R5cGUiOiAiSEQiLCAia2V5IjogInhpRTE0SWtNSVJLeFNQQlFMK0hrc0E9PSJ9LCB7Im
tleV9pZCI6ICJORGN6TVdOaE56ItOWExYy01YjQ1LThjMTMtMGY4MDJiMDRiMWNj
CAidHJhY2tfdHlwZSI6ICJBVURJTyIsICJrZXkiOiAieTlERG0zMHR0VDA3R3FYZnhLTXdkUT09In1d
LCAidXNlX3BvbGljeV9vdmVycmlkZXNfZXhjbHVzaXZlbHkiOiB0cnVlLCAicHJvdmlkZXIiOiAid2l
kZXZpbmVfdGVzdCIsICJjb250ZW50X2lkIjogIldVOVZVRVZEUlRwRGd6SzNiU0lUVGNN01BPT0iLC
AicG9saWN5X292ZXJyaWRlcyI6IHsibGljZW5zZV9kdXJhdGlvbl9zZWNvbmRzIjogMzYwMDAsICJjY
W5fcGxheSI6IHRydWV9LCAicGF5bG9hZCI6ICJDQUVTeGdzSzR3b0lBUkkxQ1FxeEFnZzNOFaE5EYUhK
dmlXVkRSRTB0VEdsWRYZ3RlRGCyR05Db2pvNEZJbzRRkTUlJQkNnN0NBUUVBNXJiVkFoYThBbVZMMUI
3OWlPMGkzeU15TFJCVJCMGNIVUJZRzTBTT3JtS2NpcWVkbms1ZSsrM3lWeHBvd0R4UzhWTVhIeUppEYUVKZE
QwV1pVZWl3SzhQQjxcxQ0ZpUlcwUEhiMDZCVThXZFtcW1Kbzlmdm1NNndUt2WVF3dlQyMkF6bGd5L2dnU"

VN5aUc0MFBOSEU3ZWVVcnZESWJ5OVVvWEtWdmQzczJMVXVVQllBOGxtd1RWaktqM0ErYmZXTmtiODZK
NjU3djF0bmxNbmNWNUkxK1FKZnMrcnB2eXRYbEIrNG40c3M0NTRUNElBMGJmN0dKSHJYeXBJWVZOZER
qeTVRaXU2c2UxV1VJVFJGZlR5ZjBYVTJzU3NyLzJVHl3NlVCdUp5V0U5V2tppd29tdFdPei9JL0pUU3
g0eVAvWkxnelpLVEVMK2Z6Z2VNdFp6ZFRyMW1OQjBBRSURBUUFBCS05jZ0VvQUNuuU1xY05MRHBncTEvc
S91Y0t3UmovdFRjc0JyMkNPSmRSMnhhJTDVlYXA4Nmo3UXNNZmwxRnRJYWZQY1lzMkRWTDI4VDJ4SnBa
QUd2VDlMK3kvb1FwaEVuxNFc2amFIYlQrNW9SNDFZcGdxaSU5aZU5oVkc4ZzIyTUhhNRGg4bjB
tNVZPNmk2bTBBFUFU4aTNacUY3UlBveVY0THJSNjdtTzR3NjFMbGFkR1lhejQvdXoyRVddIbDNEVW9sN2
hYT2hCVHNpOWZOc2VNZG5ycC83TFZaOHBTUThEeGFmaGVVZmlQSERzcHArMzlvTXFQcHdFdGdyemF4M
nNSQ3hDc2ZyMFNhRGZNWmN2aGtIbWhVRFdBM1hmTEhrREJqUmhvVS00zMFFXTjE0M1FLei9mWCtxWUFB
OXJvS2xnZE14TTAxWkNBBMFRHZE1GN0pIVkrRna1hLS0JxMEJRcXVBZ2dCRWhDRERmVOVWs4RUxrMXl4RVp
NTVpnOTRhR0wya2pvNEJZbzRDVTlJQkNNnS0NBUUVBNkRXemNocDwakFrY0U5bWVPT2p0ZmxxTNXNPQn
RoZGZKSVI3T21QL1ZzTVAvMzRJL2F5WWt6Yjh0Tm42TXJqQkZHL1I2VTM0dUlrV00ybGY1SldHQ3hZT
lVQbVhDc0hXVdvb3lFQkVHR1JuczIzWDRECEt4ODVEYWpoEcVdrR2VGGOHBJQldwa01KaEEk0V3dLNkJo
MCtJSng2UGJyVStzYjNYUTRqcHBVWm55ci9nWVZRNkRXRmJ5KzFUWXNKN0RsUDBHTWZ0Z0FwTUFSWWt
Qa1NsY2VwSlBrU01pUGRzSllvQUVYK2lIMGFnWkZBZmtyYjRsTThxUmJd2xrTUx0Q0E4aXdkQU0hlWk
lKTnpJc1E4WDhxMFRkdFFlb3VveXlVRndkcmdPeBeEtiWUpGaXl6UVF3ZURib0xWVm9JK1ZZN3VyVFdhO
Wd2V2U1eTRRR3phQjFJeDBBRSURBUUFBCS05jZ0VvQURuuy9nK2JqUlZjjOWcxZG9PZVzVScENEST0g0ZnNF
M0NyR3BzZ0VBSnA2NFdrdFVTFsQk9GdXXkvQ1yyY1Z4YVVWMWlQ3cEJLM2c5NU1qSVg3OG40eGhPWDM5SVR
5a3Juc09WdTNNWFA4Mmo4NkZoV1lzbEJ5ck5KOHVhZ0xxwVGQra3ExbHRRZFZ2aE5rOHlV3BFWDNjeEA1aF
dDampDTFUySDAxT3FONWZNWG4zSHByMFdqdV05CQmF6cnExWWY3Y3pUbkJ4VEpBUTFzWWVVYZEo5cVAwZ
HFDakctY0FBR2tzczVxWXJZ0K053Q2hDd1Q2R3REemRmaVVsS1ZOWVFIamNSdTlBdEJzRFJUUZy9zUVRW
YW9Nc0ZoaVM1c3BYaEloXd6UTNwdXFSbFNEN01KcHBrc3llbVRwWm0vZmJJuOVZ4cGVnNC9oemh3aTZ
1Z3c0Qk8wdFN3dWRSdFNaNW5SWm5Dd0RmSWgrUHJpWVpCQWFvSlpzajgyhpSEJKc1ZNZDDkvNUI3TF
N2SkIrSTJUUjRWWSGNpdStEZZXQ2VWZla3VrZXdtanB3eDRUUjR1B0cVRJLzVaVXV5N0VNNnJRdUxCUlcvN
DB6UWRxWDNuZEhqWVpDdmlpUM3Zxd3I2cUZtT0hpTmVRMW1Idk5iemczR1VEUXxxMVZpT0ZBUUNTWHp3
eTJFd3kyUm80NUdoc0tFV0Z5WTJvcGRHVmpppSFZ5WlY5dVlXMWxxZF1o0T0RZdE5E5qUWFGZ29NWTI5dGN
HRnVlVjl1WVcxbEVnWkhia2hiMjluYkdVWUZ3b0tZWV5dVlYXdXZ1XmMuJlRzJoJenVkJJbU1YxUMFJOR2hZS0
RYQnNZWEJtYjNUMdFgyNWhiV1VT0lV4cGJuVVjRNZ2dJQVBCQmkkBRWdBQmxwVQ2xJS1BBZ0JFaERsUWg1d
W5jjMWF3SlVnTHROROMMc0TE5HzzEzYVdSbGGRtbHVaVjkwWlhOZE1lRlVSk5VWWDBOUFRsUkZmdFJmU1VS
Zk1Tb0NNMFFUBJQkRkoQnVWVCsxRGJ4dUkzc3FNFUzJWWGZDU0dBRWd6ZVM2bmdVd0ZScUFCcG5SVTR
iUkV0OU9JZzhQZHJFSU0wRVU4NUFUbHczZVI5VU42Vm5VVUw4UWdERM0RBWEx4blZxT0I4V1pCVEJ5a3
pKTDRIUVZtQzhQcW5wYUxGWHlrSXZMcFBTb200WGhsNDVSSUJwNzRkTkddON21YcTlNS2xmeG9DTlJ1c
UVSVkRndzFzMThSZlh2QVNtbng2Z2ViUjVmaDg3dmZ2R2R1TTG9TamhEUUddINGpoSXRWK24yoSkJxR1BY
ZWEyaXFGdVVyZmlhem82ZTRLMnpsMTJqQ1hBMVJPUdZytoeEhscURvdoydDRvRXFCdVlVzvhVRkg
0WU1ucUxxlOHB0V0dRMDBVR1gvcUY5K1pTUnpnU1RLTGd0V0plTStTMGNOSUtnSEZUWmkvSmNaU0N5Vm
tzY0VqQW5pTVU5a1BqQkkj6YlIzTHJQZXZ2WFNkZmFOUVWVUNiU2U4PSJ9", "signature":
"VYLAAktBcg3nAVexs/owBk95mNCs7a9ATVNlu+E2nj8="}'

https://license.uat.widevine.com/cenc/getlicense/widevine_test

Expected Response

{"status":"OK","license":"CAISyAMKKgoQblU/tQ28biN7KjEtlV3wkhIQblU/tQ28biN7KjEtl
V3wkhoAIAEoABIKCAEQARgBMKCZAhpmEhAnLYoeF1JERtTGJn3PZ43xGlAuk4wxrzu3jBux9utLoVA2
AgxlTskglmzmVxZxqsz9CgIq24i3DSRJAblpFscyG4bg1SEAyOCWBCMKsNZ5leVnADRnVH20gfwoj/u
c4iz8FSABGl4KJDBkZjI1Y2M1LTlhZDEtNTI4Yi05MGRjLTU2MmNkMTgxMTgzYhIQd9H5HL/xbXCSnL
bjrsxZ4Rog0b01gbD9VbfpmklAQol7oAC1/wZpXNie/BqJniPehBAgAigBGl4KJDc1ZWE4YmFjLWY4N
2MtNWMzMS1hZmE2LTVmN2JlMDYxNTBlORIQcKCpMlX8UpJ1lazb6QAyehogckuIGdK43wCoChZ8qlpX
J+6fsRgnkC0rLg/hx4Kn2oYgAigBGl4KJDQ3MzFjYTcyLTlhMWMtNWI0NS04YzEzLTBmODAyYjA0YjF
jYxIQMszrg257yeHnZH71T11g6Rogys15dHsuh8M+H0LpJORM5JLRzQ6Q6t9i+qJDxVfsfjkgAigBIM
3kup4FOAAaIG2P7dWmCAY2a59Vu/HGJ8IbLbqA0UQ0yUETDC0VuJIKIoAC019cqv20XIGOmbAJPLAhh
elyBDTYXkWuzk3oZX7G3tbbGUgHBBVoVXyMGiqr2ZTPKsc7EfPnofdMhgfdtomKr/ezsQM9TWAeKy8s
9X04cBG42/N7CtR2186WxJdwA2E+uipptJKBzEpB9huAKbGpxQLA5onli6tBn3ft7qz3fPAWAr9Jkpj
iCIcqosXM03A6n1AQP6GvAPXN2S9aiy28JacHQCyE97OLjAL9yiG9MuTIvEBxdG/cQ5wloOTdK2ICGF
q3vEOfLfVCqMvgumKQ4QMXZmuH/NUMIRT89Z8Ibq4wHo1nFMjyrKFpmkj57Xuh3w7CzCEWnD43rp+IM
tN1qQ==","license_metadata":{"content_id":"WU9VVFVCRTpDd3E3bWVXZWM3MA==","licen
se_type":"STREAMING","request_type":"NEW"},"supported_tracks":[{"type":"SD","ke
y_id":"MGRmMjVjYzUtOWFkMS01MjhiLTkwZGMtNTYyY2QxODExODNi"},{"type":"HD","key_id"
:"NzVlYThiYWMtZjg3Yy01YzMxLWFmYTYtNWY3YmUwNjE1MGU5"},{"type":"AUDIO","key_id":"
NDczMWNhNzItOWExYy01YjQ1LThjMTMtMGY4MDJiMDRiMWNj"}],"make":"Google","model":"Ch
romeCDM-Linux-x86","security_level":3,"internal_status":0,"session_state":{"lic
ense_id":{"request_id":"blU/tQ28biN7KjEtlV3wkg==","session_id":"blU/tQ28biN7KjE
tlV3wkg==","purchase_id":"","type":"STREAMING","version":0},"signing_key":"9Y+m
1Rf4AmqWRT01EpagStZ/zREV4Dz8es611yBiZ5wIo0wEQ4FfLNHwTnq2shN218OlOx3reeR/+R05ltH
/wg==","keybox_system_id":4183,"license_counter":0},"cert_serial_number":"NDM2O
DcyNmY2ZDY1NDM0NDRkMmQ0YzY5NmU3NTc4MmQ3ODM4MzY=","device_whitelist_state":"DEVI
CE_WHITELISTED","message_type":"LICENSE","platform":"","device_state":"RELEASED
","pssh_data":{"key_id":["5QB+bp3NWsCVIC7TdYOCzQ=="],"content_id":"VEVTVF9DT05U
RU5UX01EXzE="},"client_hdcp_version":"HDCP_NONE"}

Offline License

Two types of licenses exist, streaming and offline. A streaming license is used in most cases where the content is streamed to the device and is viewed as it is streamed. This type of license is loaded in memory and is never persisted on the device. Once the session is closed on the device, the license is cleared from the device.

The license type (STREAMING or OFFLINE) is specified in the license request on the client. For example on Android, *MediaDrm.getKeyRequest()* takes a *key_type* parameter to indicate STREAMING or OFFLINE. A DRM license will not persist unless the application specifies OFFLINE as the *key_type.*

An offline license is used where the content is stored on the device for later viewing, generally in cases when the device will be offline. This type of license is persisted on the device and can be loaded at a later time without contacting a license server. An offline license remains on the device until it is released.

The license server proxy must indicate that the license can persist on the device by setting '*can_persist=true*' in the license policy. The license server should reject an OFFLINE license request if it does not want the license to persist.

License renewals should not set can_persist to true as shown below.

```
License.Policy playback_policy;
playback_policy.set_can_persist(true);
playback_policy.set_can_renew(false);    // This is the default.
```

Offline License Release

Once a license is persisted on the device and the client wants to remove the license, the application will initiate a license release. This will generate a license request with the request type (NEW, RENEWAL or RELEASE) set to RELEASE.

When handling a license where the request type is set to RELEASE, the policy's *can_play* attribute must be set to false as shown below.

```
License.Policy playback_policy;
playback_policy.set_play(false);
```

The Widevine client will not delete the license from the device until it receives a valid license response indicating the license release was handled by the license server. The proxy must return the license to the client.