WIDEVINE®

CYPHER FOR DIGITAL MEDIA

PROXY INTEGRATION

VERSION: 1.9

Widevine Technologies

901 5th Ave, Suite 3400

Seattle, WA 98164 USA

www.widevine.com

206.254.3000 voice

206.254.3001 fax

sales@widevine.com

WIDEVINE®

CONTENTS

Revision History

Version	Date	Description	By
1.0	08/30/2010	Updates for 4.4.5 release	Alex Lee
1.1	10/11/2010	Updates for 4.5.0 release - Added storefront value	Alex Lee
1.2	11/01/2010	Added setpurduration	Alex Lee
1.3	11/09/2010	Updated Terms and abbreviations	Alex Lee
1.4	01/20/2011	Additional custom error code range Additional custom portal override fields - Setofflineplayback - Setonlinestreaming	Alex Lee
1.5	03/15/2011	Removed offlineplayback, onlinestreaming Added playbackmode Added signing procedure	Alex Lee
1.6	03/17/2011	Added denyhd, usageurl	Alex Lee
1.7	04/12/2011	Removed playbackmode	Alex Lee
1.8	04/20/2011	Clarification on Purchase Duration requirement	Alex Lee
1.9	06/25/2011	Updated sample proxy cgi	Alex Lee

1. Purpose

The purpose of this document is to describe the Widevine license proxy component, including integration into an existing customer's authentication portal service.

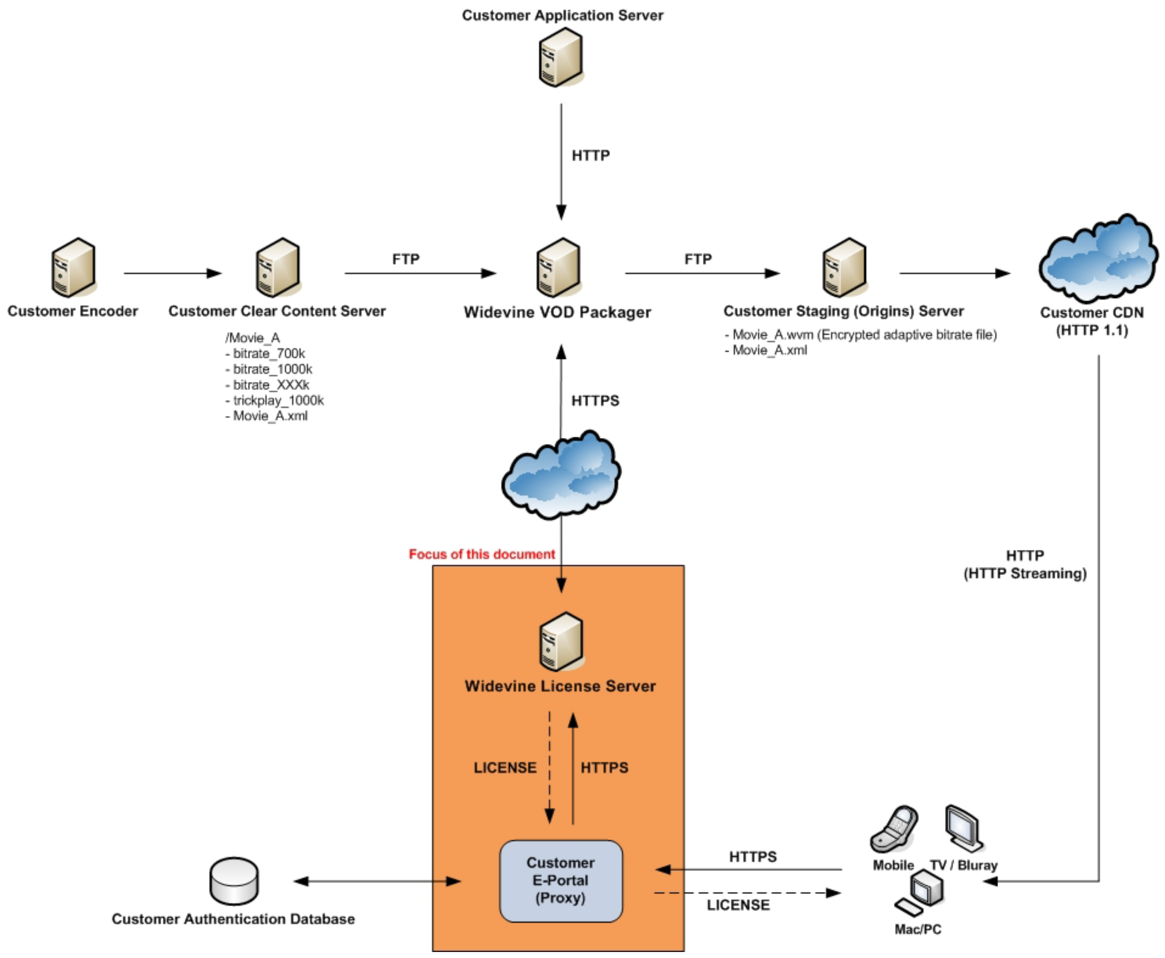

FIGURE 1 – CYPHER COMPONENT DIAGRAM

1.1. Customer E-Portal (Proxy)

The e-portal is the storefront.

The proxy, which will likely reside on the e-portal, allows the service provider to leverage their consumer information to implement business rules. The proxy is a common gateway interface (cgi) that Widevine provides to the service provider. The cgi is written in Perl, however may be re-written in another language if the service provider so chooses provided the cgi protocol is used. The cgi will be hosted on a service provider provided HTTP server and will receive HTTPS license requests from the Widevine client which can be interrogated and either forwarded to the license server or declined with a message bubbled back up to the end-user.

1.2. Customer Authentication Database

This represents the service provider's data store which contains information used to implement business rules.

1.3. Widevine License Server

This represents the DRM license servers. The license servers are hosted by Widevine in the cloud.

2. Terms and abbreviations

- **Asset -** Physical media such as a H.264-encoded movie
- **Asset Name –** A valid name of the asset that must be unique when combined with the Asset Owner.
- **Asset Owner –** The entity that is encrypting or packaging the asset. A Widevine-provided System ID is required for all Asset Owners.
- **Asset Provider –** The entity that will be providing the asset to the end user. This is identical to the portal or license proxy (id field) that is requesting a license. A portal will only obtain a license if it is listed as an asset provider for that asset.
- **Catcher** - A temporary storage device used to collect on- demand files and request missing parts (due to unreliable transport) prior to provisioning of content to the video delivery servers
- **Client device -** the equipment used by consumers to view digital media.
- **Container -** A container format is a computer file format that can contain various types of data, compressed by means of standardized audio/video codecs
- **Container -** A container format is a computer file format that can contain various types of data, compressed by means of standardized audio/video codecs
- **Content –** A collection of assets and metadata files
- **Digital Rights Management (DRM)**-security software to allow a Service Provider to implement business rules which control a consumer's access to digital media.
- **E-portal –** Refer to License Proxy.
- **License –** Encrypted data that contains the viewing rights associated with digital media and the key to unlock the content key. Each license is generated dynamically and individualized to the requesting client. A license request may only originate from a Widevine client.
- **License Duration –** The amount of viewing / playback time (content decryption)
- **License Proxy –** A software component that performs the business logic (authorization and validation) functions prior to making a license request from the Widevine License Server.
- **License Server –** Also known as key server. The Widevine License Server maintains the secure storage of all encryption information as it pertains to assets. It also maintains the policies (business rules) that are managed by a provider. The primary function of a license server is to issue licenses (governed by configurable business rules) when requested by authorized entities.
- **Portal –** Refer to License Proxy.
- **Purchase Duration –** Commonly referred to as rental window or license validity window. This refers to the allowed grace period in which playback is permitted. This value should be greater than the license duration value or non-zero.

3. Description

A service provider can control access to content leveraging the business rules in their back-end. This is done by using the concept of a proxy.

4. File locations

Proxy script in CGI (Perl)

- 4.5.0
  - http://wstfcps005.shibboleth.tv/demo/450/proxy.4503913.tar.gz

5. Installation

The proxy is a simple program that Widevine provides to the service provider to allow the license request from the client to be intercepted so business rules can be implemented. The program that Widevine provides is a Perl script called GetEMMs.cgi. The Perl script has the following dependencies (based on a RHEL5 OS):

- httpd
- Perl
- perl-Compress-Zlib
- perl-HTML-Tagset
- parser-HTML-Parser
- perl-libwww-perl

As well as a recommended cgi-bin location of **`https://<host>/widevine/cypherpc/cgi-bin`** , where GetEMMs.cgi will be located as that is the location where the DRM client issues a license request.

To configure the proxy Widevine will provide an archive portalProxy.tar.gz. When unarchived there will be the following directories:

- GetEMMs.cgi – the proxy Perl script
- rhel4 – dependencies for RedHat Enterprise Linux 4
- rhel5 - dependencies for RedHat Enterprise Linux 5
- widevine.conf – httpd configuration script to change cgi-bin location
- README.Setup.html – html subset of instructions from this document

The request from the client will arrive as:

`https://<host>/widevine/cypherpc/cgi-bin`

which requires the Widevine httpd.conf be used. To install the conf file (example based on Red Hat Enterprise Linux using Apache):

1. Install the dependencies
2. Copy `/var/www/html/PortalProxy/widevine.conf` into `/etc/httpd/conf.d/`.
3. Restart httpd (`/etc/init.d/httpd restart`)
4. Copy `/var/www/html/PortalProxy/GetEMMs.cgi` into `/usr/local/widevine/cypherpc/cgi-bin/`. (If the destination dir does not exist, use 'mkdir -p' to create it.)

The Perl script can be used as is, or can be re-written in another language. In this model, the e-portal receives the request from a client to view content. When the request comes in it is intercepted by the proxy so the service provider can verify if the user has paid, has a subscription etc. Then the request can be forwarded to the license server or declined by the service provided with a message bubbled up to the user.

After configuration, when a user plays encrypted content, the proxy will receive an asset identifier in a URL sent by the Cypher client and needs to do one of the following:

- Reject the request if the user is not authenticated.
- Reject the request if the user is not entitled to view the specified content.
- Forward the request to the license server.

The proxy is configured in the WidevineMediaTransformer.js by pointing to the proxy server in:

```
var emm_url     = "https://<host>/<path>/GetEMMs.cgi";
```

The following sequence diagram shows the sequence of events when user-based DRM is used.

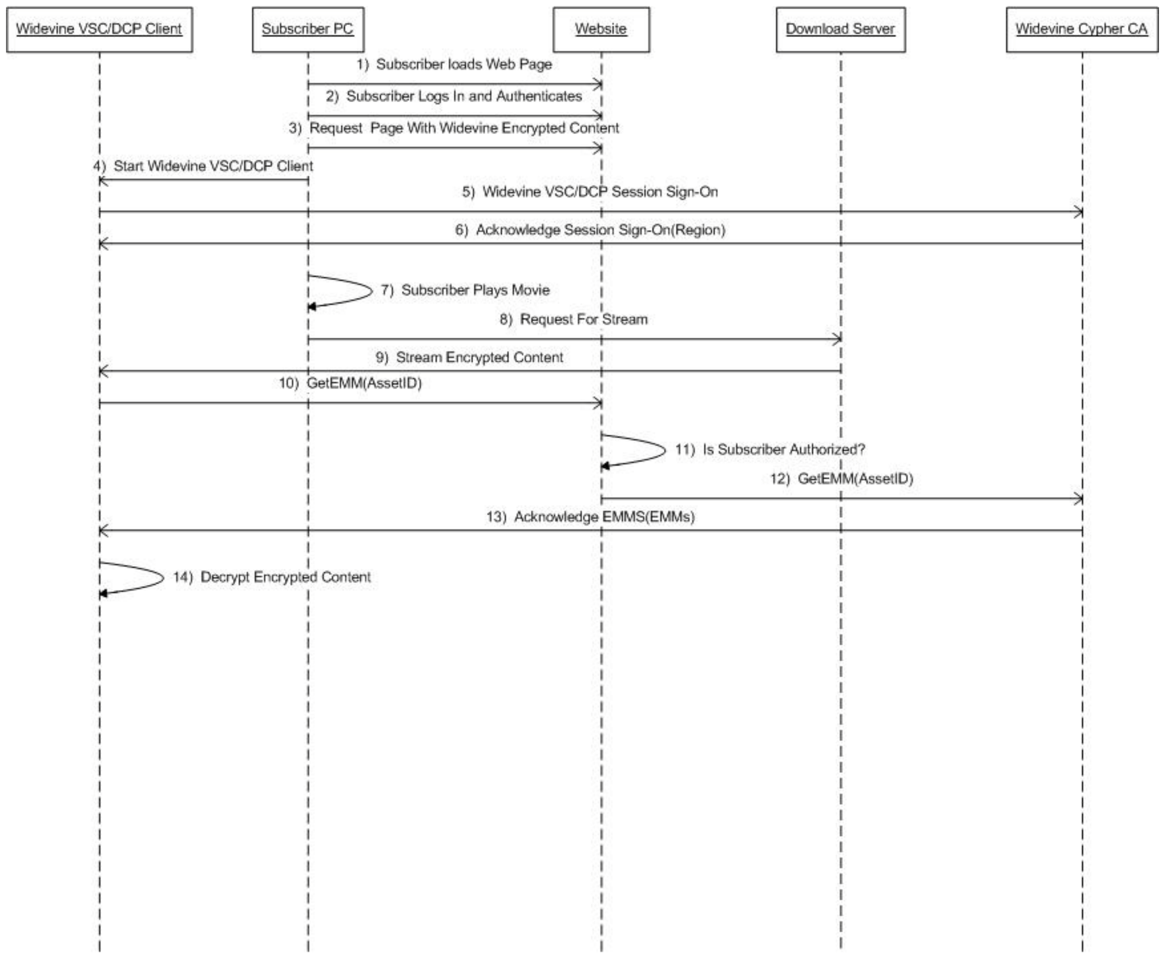
Widevine VSC/DCP Client
Subscriber PC
Website
Download Server
Widevine Cypher CA
1) Subscriber loads Web Page
2) Subscriber Logs In and Authenticates
3) Request Page With Widevine Encrypted Content
4) Start Widevine VSC/DCP Client
5) Widevine VSC/DCP Session Sign-On
6) Acknowledge Session Sign-On(Region)
7) Subscriber Plays Movie
8) Request For Stream
9) Stream Encrypted Content
10) GetEMM(AssetID)
11) Is Subscriber Authorized?
12) GetEMM(AssetID)
13) Acknowledge EMMS(EMMs)
14) Decrypt Encrypted Content

FIGURE 2 - AUTHORIZATION AND DECRYPTION EXAMPLE OF A STREAM

6. Performing the Conditional Access

The sample GetEMMs.cgi code shows how to extract the Asset Identifier and Client Identifier from the CGI requests. The e-portal may use these values to conditionally forward the GetEMMs.cgi request to the Cypher license server. The following values are available from the GetEMMs.cgi request:

CGI field	Description	Type
assetid	Unique value identifying the asset. This value is also provided during asset registration.	uint32
clientid	Unique value identifying the client.	string(32)
time	POSIX time as reported by the client.	uint32
userdata	This is a optional field and is composed of a number of subfields: **ip**: IP address of the device **streamid**: Stream ID **deviceid**: Unique device identifier **optdata**: Optional free text field The use of pipe (|) separators in custom fields is not recommended as it may cause parsing issues related to URL encoding.	string (1024)

The e-portal may insert values into the GetEMMs.cgi request to provide additional policy management. The following values may be inserted and will take precedence over values specified by the client:

CGI field	Description	Type	Required
setduration	The number of seconds the entitlement is valid. This value is in seconds and will override the configured duration value for the portal. Example: setduration=86400	uint32	Optional
setpurduration	The purchase duration in seconds. Commonly referred to as rental window or license validity window. This value must be non-zero, usually greater than the license duration value. Example of a 30 day rental window: setpurduration=2952000	uint32	Optional
denyhd	Toggle to deny HD playback Valid values are: 1 : enable (deny HD playback is true) Example: denyhd=1	int	Optional
usageurl	Specify a license usage url. Example: usageurl=<url_value>	base64 encoded string	Optional

hburl	The heartbeat URL to use.
	
	Example (base64):
	`hburl=L2NnaS1iaW4vSGVhcnRiZWF0LmNnaQ==`
		base64 encoded string	Optional

hbint	The heartbeat interval in seconds. This value must be greater than zero if the asset is using a policy indicating heartbeats are required.	uint32	Optional

	Example: hbint=60		
setpolicy	All assets are associated with a policy. The policy contains rules specific to the asset such as duration of the license. The policy is assigned to an asset during the encryption process. The portal can insert into the EMM request the name of an alternate policy to use for this instance of the license. The policy must already exist otherwise the EMM request will fail. Example: setpolicy=new_policy	string (1024)	Optional
minbr	Minimum bitrate to allow on playback. This value is in bytes per second. Example: minbr=500000 NOTE: Client functionality currently disabled.	uint32	Optional
maxbr	Maximum bitrate to allow on playback. If used, the Widevine client will limit the maximum bitrate for streaming to the value specified. This value is in bytes per second. Example: maxbr=2000000 NOTE: Client functionality currently disabled.	uint32	Optional
sf	The Storefront value (provided by Widevine). This identifies the source of the request for Widevine billing purposes. Example: sf=wvstorefront	string	Required

7. Inserting User Data into the License Request (on the Client Device)

On the client device, Widevine provides the following javascript APIs to set fields that will be used to build the userdata field in the GetEMMs.cgi request.

All functions below are for customer use and not required or generated by Widevine unless otherwise specified.

Function	Description

getClientId	Returns the Widevine client Id (16-byte UUID)
	A sample clientId is:
	`550e8400-e29b-41d4-a716-446655440000`
	
	`String getClientId()`

setDeviceId	Set the device Id. Customer can derive a unique value from getClientId() as the deviceId with optional modifications. **`void setDeviceId()`**
getDeviceId	Returns the device Id. **`String setDeviceId()`**
setStreamId	Set the stream Id. A value uniquely identifying the current stream. **`void setStreamId()`**
getStreamId	Returns the stream Id. **`String getStreamId()`**
setClientIp	Set the IP address of the device. **`void setClientIp(string)`**
getClientIp	Returns the IP address of the device. **`String getClientIp()`**
setEmmUrl	Sets the URL to use for the license request. **`void setEmmUrl(string)`**
getEmmUrl	Returns the URL for the license request.

	`String getEmmUrl()`
setEmmAckUrl	Sets the URL to use for sending and license ACK message **`void setEmmAckUrl(string)`**
getEmmAckUrl	Returns the URL for sending the license ACK message **`string  getEmmAckUrl()`**
setHeartbeatUrl	Sets the URL to use for sending heartbeats during playback **`void setHeartbeatUrl(string)`**
getHeartbeatUrl	Returns the URL used for sending heartbeats **`string getHeartbeatUrl()`**
setHeartbeatPeriod	Sets the heartbeat interval used for sending heartbeats **`void setHeartbeatInterval(string)`**
getHeartbeatPeriod	Returns the heartbeat interval **`string getHeartbeatInterval()`**
setOptData	Sets optional data field. This optional data is included in the EMM requests along with other userData values. **`void setOptionalData(string)`**
getOptData	Returns the optional data field. **`string getOptionalData()`**

8. Handling the License Request in the E-Portal

The e-portal can choose to receive the license request from the client device and do one of the following:

- Reject the request and return an license response (see GetEMMs.cgi return status)
    - As an example, for revocation/expiration of existing licenses, set the duration to 0.
- Forward the entire license request to the Widevine server. The response from the Widevine server will return back to the e-portal and the e-portal should forward the license response to the client device.
- Modify the license request and forward to the Widevine server. The response from the Widevine server will return back to the e-portal and the e-portal should forward the license response to the client device.

9. GetEMMs.cgi URL Example

```
https://wstfcps005.shibboleth.tv/widevine/cypherpc/cgi-
bin/GetEMMs.cgi?ver=4&mk=PC&md=Flash:IDM&ver=4&assetid=1000332&setd
uration=60
```

The base license request values such as Make (mk), Model (md), version (ver) and assetid are required to be sent to the Widevine License Server.

10. GetEMMs.cgi Return Status

The return status from the GetEMMs.cgi request is handled by the client. In the case where the e-portal receives the GetEMMs.cgi request and decides to block the request, a response from the e-portal to the client is expected. The e-portal should return a 4 byte status value in the range from 512 to 768 to indicate a 3rd party status, followed by the 4 byte asset ID. The 8-byte response should be base64 encoded.

An error response with a status value between 512 and 999 are assigned for customer use. Error responses from 512 to 768 will result in the removal of the existing stored license (if it exists). Error responses from 769 to 999 will not remove the existing stored license (if it exists).

An error response to the GetEMMs.cgi request where the status value is 512 and Asset ID 1000293 is as follows:

```
HTTP/1.1 200 OK
Date: Mon, 24 Nov 2008 22:10:28 GMT
Server: Apache/2.0.52 (Red Hat)
Length: 12
Encoding: base64
Connection: close
Transfer-Encoding: chunked
Content-Type: text/plain; charset=ISO-8859-1
12
AAACAAAPQ2U=
```

10.1. GETEMMS.CGI RESPONSE FORMAT

The format for the response to GetEMMs.cgi is a Base64 encoded sequence of bytes in network byte order:

➢ License status	(4 bytes, unsigned integer)
➢ Asset Id	(4 bytes, unsigned integer)
➢ Key Id	(4 bytes, unsigned integer)
➢ EMM	(sequence of bytes)

WIDEVINE LICENSE STATUS

Value	Description
0	Unknown
1	OK, successful
2	Server-side database error: Asset not found
3	Servers-side database error: Asset save failed
4	Server-side database error: Asset delete failed
5	Server-side database error: Asset already exist
6	Internal error
7	Operation is not allowed
8	Asset is currently blocked (see access criteria settings)
9	Asset is outside license window
10	Asset is outside allowed geographic region
11	Signature is missing in the EMM request
12	Signature is not valid in the EMM request
13	Provider is unknown
14	Network error
15	EMM format error
16	EMM decode failed
17	Client network error
18	Request aborted
19	Client key is missing
20	Registration server is not responding
21	Registration server is down
22	Portal information is missing
23	Portal is unknown
24	Asset Id is missing
25	Owner information is missing
26	Provider information is missing
27	Name is missing
28	Invalid CCI
29	Invalid DCP
30	Invalid license window
31	Policy not found

32	License request rejected due to Policy
33	Policy server is not responding
34	Error processing token sent by client
35	Invalid geo-graphic region
36	Invalid Nonce
37	Invalid Hardware Id
38	Invalid Token
39	Invalid asset Id
40	Invalid name
41	Invalid client diversity
42	Invalid Key Id
43	Model not supported
44	Invalid keybox system Id
45	No device license is available
46	Unknown code
47	Invalid access criteria
48	Geographic region is missing
49	Key verification failed
50	Hash in client token failed
51	Unable to get key for client token verification
52	Wrong system Id sent by client
53	Revoked client version
54	Client version tampered
55	Client version missing
56	Asset provider already exists
57	Missing diversity information
58	Missing token
59	Client model tampered
60	Asset key is too large
61	Failed decryption
62	Too many assets
63	Make not supported
64	Policy already exists
65	Invalid XML
66	Provider violation
67	Portal verification failed
68	Portal override not allowed
512 – 768	3rd party-defined error codes (remove existing stored license, if any)
769 – 999	3rd party-defined error codes (retain existing stored license, if any)

11. ACK

If configured, an HTTP(s) acknowledgement message is sent from the client to the e-portal upon successful receipt of an entitlement. The ack message is a CGI request message sent to the same URL as the message being acknowledged.

```
http://<portalurl>/widevine/cypherpc/cgi-
bin/Ack.cgi?req=GetEMMs.cgi&ver=1&assetid=1000000&clientid=Xn4x4K0xKJKYU0F94X6s
b-
lSyvbeZCbi&status=0&userdata=ip:192.168.100.100,streamid:100,deviceid:10a50fg,o
ptdata:some%20data
```

The ack request has the following values (all in network byte order):

CGI field	Description	Type
Req	CGI request associated with this ACK message.	string(32) – GetEMMs.cgi
Ver	Request version	Int32 – license message version
Assetid	In the case of GetEMMs.cgi, this is the Widevine internal asset ID.	uint32
Status	The status received in response to the entitlement request. This will be 1 (one) as an ACK is only sent on success.	uint32
Clientid	Unique value identifying the client.	string(32)
userdata	In the case of GetEMMs.cgi, this is the User Data field sent in the GetEMMs.cgi request this acknowledgement applies to. This is composed of a number of subfields: **ip**: IP address of the device **streamid**: Stream ID **deviceid**: Unique device identifier **optdata**: Optional free text field	string (1024)

11.1. ACK RESPONSE

An HTTP response to the Ack message is expected. The response is ignored by the client.

12. Signing the License Request (Optional)

1. Append the following strings from the license CGI request and create a new string used as hash input.
    a. assetId, clientId, mk, and md
2. Append the current time in seconds (since Jan 1, 1970) to the hash input string, this is the portal time.
3. Generate a SHA-1 hash of the hash input string.
4. AES encrypt the hash, please use 256/CBC, padding is enabled.
5. Base64 encode the encrypted hash, this is the signature.
6. Add the following two name/value pairs to the license CGI request that is forwarded to the WV server:
    a. "sign=<signature>&ptime=<portalTime>" where
        i. <signature> is the Base64 encoded string
        ii. <portalTime> is the unix time used as input into the hash, represented as a string

13. GetEMMs.cgi Sample

```
#!/usr/bin/perl
# start code
use LWP ;
use CGI ;
use MIME::Base64;

$ENV{"REQUEST_METHOD"} =~ tr/a-z/A-Z/;
$query = new CGI;
my $allow         = "yes";

# Error codes less than 512 are reserved for Widevine
my $badStatus     = 600;

my $url = "https://wstfcps005.shibboleth.tv/widevine/cypherpc/cgi-
bin/GetEMMs.cgi";

open STORE_FILE, ">/tmp/GetEMMs.vars" || die "Unable to open /tmp/GetEMMs.vars
for writing";
my $assetid = 0;
if ( $allow eq "yes" )
{
         my $ua = LWP::UserAgent->new;

# --      get values from the license request
          my $mk              = $query->param('mk');
          my $md              = $query->param('md');
          my $ver             = $query->param('ver');
          my $userdata        = $query->param('userdata');
          my $messageid       = $query->param('messageid');
          my $token           = $query->param('token');
          my $divinfo         = $query->param('divinfo');
          my $extra         = $query->param('extra');
          $assetid          = $query->param('assetid');
          my $clientid      = $query->param('clientid');
          my $sessionid     = $query->param('sessionid');
          my $clienttime    = $query->param('time');

          my %formData = $query->Vars;

# --      add values to the license
         my $hburlValue = "L2NnaS1iaW4vSGVhcnRiZWF0LmNnaQ==";
         my $hbintValue = "120";
         my $ackurlValue = "L2NnaS1iaW4vQWNrLmNnaQ==";

          $formData{"hburl"} = $hburlValue;
          $formData{"hbint"} = $hbintValue;
          $formData{"ackurl"} = $ackurlValue;

# --      Forward license request to Widevine
         $response = $ua->post( $url, \%formData );

# --      Log the request variables
          print STORE_FILE "REQUEST\n";
         print STORE_FILE "=======\n";
          print STORE_FILE "mk = $mk\n";
          print STORE_FILE "md = $md\n";
         print STORE_FILE "ver = $ver\n";
         print STORE_FILE "assetid = $assetid\n";
         print STORE_FILE "clientid = $clientid\n";
         print STORE_FILE "sessionid = $sessionid\n";
         print STORE_FILE "userdata = $userdata\n";
         print STORE_FILE "messageid = $messageid\n";
         print STORE_FILE "token = $token\n";
         print STORE_FILE "divinfo = $divinfo\n";
         print STORE_FILE "extra = $extra\n";

         print STORE_FILE "ackurl = $ackurlValue\n";
         print STORE_FILE "hburl  = $hburlValue\n";
         print STORE_FILE "hbint  = $hbintValue\n";

# --      Forward the license response from Widevine back to the client
         print $query->header(-length=>$response->content_length,
                    -type=>$response->content_type);

         print $response->content;
         $decoded_response = decode_base64( $response->content );
}
else
{
        # NOT authorized to play this content
        $assetid        = $query->param('assetid');
       $badResponse   = pack "NN", $badStatus, $assetid;
       $response = encode_base64($badResponse);
       print $query->header(-length=>length($response),
                    -encoding=>base64,
                    -type=>"text/plain");
       print $response;
       $decoded_response = decode_base64( $response );
}
print STORE_FILE "\n";
```

```
# bytes 0 through 3 contain response status code
# parse response status code
for ( $i = 0; $i < 4; $i++ )
{
  $response_byte = sprintf("%02X", ord(substr($decoded_response, $i, 1)));
  $response_status .= $response_byte;
}
$response_status_dec = hex($response_status);

# bytes 4 through 7 contain response asset id
# parse response asset id
for ( $i = 4; $i < 8; $i++ )
{
  $response_byte = sprintf("%02X", ord(substr($decoded_response, $i, 1)));
  $response_asset_id .= $response_byte;
}
$response_asset_id_dec = hex($response_asset_id);

print STORE_FILE "RESPONSE\n";
print STORE_FILE "========\n";
print STORE_FILE "status = $response_status_dec, asset id =
$response_asset_id_dec\n";
close STORE_FILE

# end code
```