Using the Shaka Packager and Player

Version	Date	Description	Edited by
0.1	5/26/2015	Initial draft	Alex Lee
0.2	7/11/2016	Corrections	Alex Lee
0.3	11/9/2016	Minor revisions	Alex Lee

Overview

The following set of exercises is designed to guide you to:

- Use of the Widevine shaka Packager SDK
- Packaging and Encryption
- Perform media analysis
- Encrypt content and generate playlists
- Use a license proxy for cloud license service access
- Playback of encrypted content

Typographical conventions

The following typographical conventions are used in this lab exercise document:

- `Courier New`
  - This font is used to display all commands and code snippets. All code samples will be highlighted with a gray background as shown below.
- Parameters that need to be replaced by specific values are included in angular braces and highlighted in yellow.
  - Angular brackets need to be excluded from the command.
- Your `<Server IP address>` is your Linux server.
- Note that **$** is not to be re-typed, it is the command prompt.
- Multi-line command syntax is indicated by a trailing backslash (\ ).

Requirements

To execute the steps in this document, it is required to use Ubuntu Linux (recommended to use the latest LTS version). For the sake of simplicity, all commands are executed as `user=root`.

To ensure you have the basic build environment, run the following commands as `user=root`:

```
# apt-get install -y build-essential gcc git g++ apache2 wget
openjdk-7-jre-headless subversion
```

It should also prompt to install any additional dependencies, say `Yes`.

Credentials

For the purposes of this exercise, we will use the `widevine_test` credentials in the test environment.

```
Provider / Signer = widevine_test
IV = d58ce954203b7c9a9a9d467f59839249
KEY = 1ae8ccd0e7985cc0b6203a55855a1034afc252980e970ca90e5202689f947ab9
```

Install EME Logger Chrome extension

This will allow you to have better logging capabilities within Chrome browser when executing encrypted content playback using EME.

Go to Chrome Extensions:
https://chrome.google.com/webstore/category/extensions

In the left search box, search for “EME Logger”

EME Call and Event Logger

from EME Logger Admins

https://chrome.google.com/webstore/detail/eme-call-and-event-logger/cniohcjecdcdhgmlofniddfoeokbpbpb/related

Install the extension.
To use the logger, click on the Logger puzzle icon and check the box “***Log to a separate frame***”
Refer to this EME logging window at the time of content playback to observe EME events. You may leave this window open at all times.

Build the Shaka Packager

1. Packager source is located at https://github.com/google/shaka-packager
2. Pull gclient and ninja from Chrome Depot Tools:

```
$ git clone https://chromium.googlesource.com/chromium/tools/depot_tools.git
```

3. Add depot_tools to your PATH:

```
$ export PATH=$PATH:`pwd`/depot_tools
```

   a. Note that the above command contains a`(back quote), not to be interpreted as a ‘ (single quote).
   b. You may want to add this to your .bashrc file or your shell's equivalent so that you don’t need to reset your $PATH manually each time you open a new shell.

4. Run the following commands in sequence.
   Note: Ignore the warning messages about being sad.

```
$ mkdir shaka-packager
$ cd shaka-packager
$ gclient config https://www.github.com/google/shaka-packager.git
--name=src
$ gclient sync
```

5. Run the following to verify the directories were created:

```
$ ll src/
```

6. Build using ninja to create the reference binaries for use. This takes a while, grab a cup of coffee.

```
$ ninja -C src/out/Release
```

7. Verify the binaries exist. Look for `packager` and `mpd_generator`.

```
$ ll src/out/Release
```

Getting Started

Navigate to the Release directory

```
$ cd ~/shaka-packager/src/out/Release/
```

The following binary files will be used:

a. **packager**
   i. Used to analyze and encrypt media files
b. **mpd_generator**
   i. Used to generate playlist files

For ease of use as you will be running these test binaries often, add the Release directory to your PATH:

```
$ export PATH=$PATH:$HOME/shaka-packager/src/out/Release
```

You now have a working packager setup and are ready to process and encrypt content.

Getting Help

Use the following command to review available packager commands

Run:

```
$ packager --help
```

Prepare files

All commands to be executed as **root**.

1. Copy the following files to your Ubuntu Linux under `/root`:
    a. Sample content

```
$ wget https://storage.googleapis.com/wvtraining/content.tgz
```

2. Decompress the tgz files to `/root`:
    a. Run

```
$ tar xzvf content.tgz -C /root
```

    b. This will create the following subdirectories:
        i. content

Shaka Player Setup

The Widevine Shaka Player is an open-source JavaScript library which implements a DASH client. It contains a HTML5 player demo that we will use for playback.

1. Navigate to the html directory

```
$ cd /var/www/html
```

2. Run the following command to download the shaka player

```
$ git clone https://github.com/google/shaka-player.git shaka
```

3. Navigate to the “shaka” directory

```
$ cd shaka
$ git checkout v1.6.5
```

Note: For the purposes of this exercise, we use an older version of the Shaka Player. We recommend using the latest Shaka Player.

4. Build player using

```
$ ./build/all.sh
```

5. Navigate to `http://<Your Server IP Address>/shaka/` in your web browser to open the Shaka player.
    a. This is your Shaka Player page.
    b. You will use this page for testing playback of any content.

Content encryption

This section provides instructions to perform the following:

- Analyze media files
- Encrypt audio and video files using the shaka-packager reference binaries
- Generate playlist file that describes the encrypted files
- Playback the encrypted content

Media File Analysis

In order to analyze your audio / video stream, you can use the `--dump_stream_info` flag.

Run the following command **as root** to analyze stream info in test clips:

```
$ packager input=/root/content/750_tears_of_steel_base_480p.mp4
--dump_stream_info
File "/root/content/750_tears_of_steel_base_480p.mp4":
Found 1 stream(s).
Stream [0] type: Video
 codec_string: avc1.42c01e
 time_scale: 12288
 duration: 9021440 (734.2 seconds)
 language: eng
 is_encrypted: false
 codec: H264
 width: 720
 height: 300
 nalu_length_size: 4
```

Understanding the output:

`Found <#> stream(s)`	`Number of streams found in the content file`
`Stream [0] type:`	`Specifies the stream number and if it is video or audio`

codec_string:	<Insert info>
time_scale:	The number of time units that pass per second in its time coordinate system. A time coordinate system that measures time in sixtieths of a second. See ISO-BMFF spec for additional details.
duration	Expressed in seconds
language:	language of the video
is_encrypted:	Determines if content is encrypted
codec:	video codec
width:	Video width
height:	Video size
nalu_length_size_:	Network Abstraction Layer Units - typically expressed in 1, 2, or 4 bytes

Audio Encryption

This step will encrypt the audio file using your cloud credentials (IV and KEY pair) for authentication.

- Generate your content ID. The ID must be in hex.
    a. Run:

```
$ echo -n <random alphanumeric string> | xxd -p
```

    b. Include this **hex output** in the command below.
- Run the following command:

```
$ packager
input=/root/content/128k_tears_of_steel_audio_eng.mp4,stream=audio,output=en
c_tears_audio.mp4 \
--enable_widevine_encryption \
--key_server_url
"https://license.uat.widevine.com/cenc/getcontentkey/widevine_test" \
--content_id "<hex output>" \
--signer "widevine_test" \
--aes_signing_key
"1ae8ccd0e7985cc0b6203a55855a1034afc252980e970ca90e5202689f947ab9" \
--aes_signing_iv "d58ce954203b7c9a9a9d467f5983924" \
--crypto_period_duration 0 \
--output_media_info
```

- Verify two files are generated
    enc_tears_audio.mp4
    enc_tears_audio.mp4.media_info

- View the media_info file.

Sample output

```
bandwidth: 132252
audio_info {
  codec: "mp4a.40.2"
  sampling_frequency: 44100
  time_scale: 44100
  num_channels: 2
  language: "eng"
  decoder_config: "\022\020"
}
init_range {
  begin: 0
  end: 1000
}
index_range {
  begin: 1001
  end: 1920
}
media_file_name: "enc_tears_audio.mp4"
media_duration_seconds: 734.13953
reference_time_scale: 44100
container_type: CONTAINER_MP4
protected_content {
  default_key_id: "\201_;;\301eT\025\210k\346\2707\367f?"
  content_protection_entry {
    uuid: "edef8ba9-79d6-4ace-a3c8-27dcd51d21ed"
    pssh:
"\000\000\000hpssh\000\000\000\000\355\357\213\251y\326J\316\243\310\'\334\32
\035!\355\000\000\000H\010\001\022\020$\366v\246\026DR\363\276\026s\330\0165\
27\300\022\020[h\371c\252\324^\267\212\270z\205\221+\305K\022\020\201_;;\301e
\025\210k\346\2707\367f?\032\005cwip1\"\005\0224Vg\2102\000"
  }
}
```

Understanding the media_info file:

bandwidth:	The bitrate of the stream
audio_info {	Specific audio information for the stream
codec: "mp4a.40.2"	
sampling_frequency: 44100	
time_scale: 44100	
num_channels: 2	
language: "eng"	
decoder_config: "\022\020\000\000"	This is a binary string representing decoder specific information.

init_range {
  begin: 0
  end: 593
}	

index_range {
  begin: 594
  end: 781
}	

media_file_name: "bourne_audio.mp4"	
media_duration_seconds: 125.10912	
reference_time_scale: 44100	
container_type: CONTAINER_MP4	Container type of the stream

Video Encryption

This step will create an encrypted video file using your IV and KEY pair for authentication.

- Generate your content ID. The ID must be in hex.
    a. Run:

```
$ echo -n <random alphanumeric string> | xxd -p
```

    b. Include this **hex output** in the command below.

- Run the following command:

```
$ packager /
input=/root/content/750_tears_of_steel_base_480p.mp4,stream=video,output=enc
_750k_tears.mp4 \
--enable_widevine_encryption \
--key_server_url
"https://license.uat.widevine.com/cenc/getcontentkey/widevine_test" \
--content_id "<hex output>" \
--signer "widevine_test" \
--aes_signing_key
"1ae8ccd0e7985cc0b6203a55855a1034afc252980e970ca90e5202689f947ab9" \
--aes_signing_iv "d58ce954203b7c9a9a9d467f5983924" \
--crypto_period_duration 0 \
--output_media_info
```

- Verify two files are generated

```
enc_750k_tears.mp4
enc_750k_tears.mp4.media_info
```

- View the media_info file

Sample output

```
bandwidth: 686572
video_info {
  codec: "avc1.42c01e"
  width: 720
  height: 300
  time_scale: 12288
  frame_duration: 512
  decoder_config:
"\001B\300\036\377\341\000\031gB\300\036\331\000\264\'\373\300D\000\000\003\0
0\004\000\000\003\000\300<X\271 \001\000\004h\313\214\262"
  pixel_width: 1
  pixel_height: 1
}
init_range {
  begin: 0
  end: 1126
}
index_range {
  begin: 1127
  end: 1902
}
media_file_name: "enc_tears_video.mp4"
media_duration_seconds: 734.16669
reference_time_scale: 12288
container_type: CONTAINER_MP4
protected_content {
  default_key_id: "$\366v\246\026DR\363\276\026s\330\0165\027\300"
  content_protection_entry {
    uuid: "edef8ba9-79d6-4ace-a3c8-27dcd51d21ed"
    pssh:
"\000\000\000hpssh\000\000\000\000\355\357\213\251y\326J\316\243\310\'\334\32
\035!\355\000\000\000H\010\001\022\020$\366v\246\026DR\363\276\026s\330\0165\
27\300\022\020[h\371c\252\324^\267\212\270z\205\221+\305K\022\020\201_;;\301e
\025\210k\346\2707\367f?\032\005cwip1\"\005\0224Vg\2102\000"
  }
}
```

Generate the MPD playlist

You will need to pass the media_info files into `mpd_generator` to generate a playlist file.

```
$ mpd_generator \
--input "enc_tears_audio.mp4.media_info,enc_750k_tears.mp4.media_info" \
--output "test.mpd"
```

Sample MPD

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Generated with https://github.com/google/edash-packager version
e0e0925-release-->
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xlink="http://www.w3.org/1999/xlink"
xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 DASH-MPD.xsd"
xmlns:cenc="urn:mpeg:cenc:2013" minBufferTime="PT2S" type="static"
profiles="urn:mpeg:dash:profile:isoff-on-demand:2011"
mediaPresentationDuration="PT734.1666870117188S">
  <Period id="0">
    <AdaptationSet id="0" contentType="audio" lang="en">
      <Representation id="0" bandwidth="132252" codecs="mp4a.40.2"
mimeType="audio/mp4" audioSamplingRate="44100">
        <AudioChannelConfiguration
schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011"
value="2"/>
        <ContentProtection value="cenc"
schemeIdUri="urn:mpeg:dash:mp4protection:2011"
cenc:default_KID="815f3b3b-c165-5415-886b-e6b837f7663f"/>
        <ContentProtection
schemeIdUri="urn:uuid:edef8ba9-79d6-4ace-a3c8-27dcd51d21ed">

<cenc:pssh>AAAAaHBzc2gAAAAA7e+LqXnWSs6jyCfc1R0h7QAAAEgIARIQJPZ2phZEUvO+FnPYDj
XwBIQW2j5Y6rUXreKuHqFkSvFSxIQgV87O8FlVBWIa+a4N/dmPxoFY3dpcDEiBRI0VmeIMgA=</ce
c:pssh>
        </ContentProtection>
        <BaseURL>enc_tears_audio.mp4</BaseURL>
        <SegmentBase indexRange="1001-1920" timescale="44100">
          <Initialization range="0-1000"/>
        </SegmentBase>
      </Representation>
    </AdaptationSet>
    <AdaptationSet id="1" contentType="video" width="720" height="300"
```

```
frameRate="12288/512" par="12:5">
      <Representation id="1" bandwidth="686572" codecs="avc1.42c01e"
mimeType="video/mp4" sar="1:1">
        <ContentProtection value="cenc"
schemeIdUri="urn:mpeg:dash:mp4protection:2011"
cenc:default_KID="24f676a6-1644-52f3-be16-73d80e3517c0"/>
        <ContentProtection
schemeIdUri="urn:uuid:edef8ba9-79d6-4ace-a3c8-27dcd51d21ed">

<cenc:pssh>AAAAaHBzc2gAAAAA7e+LqXnWSs6jyCfc1R0h7QAAAEgIARIQJPZ2phZEUvO+FnPYDj
XwBIQW2j5Y6rUXreKuHqFkSvFSxIQgV87O8FlVBWIa+a4N/dmPxoFY3dpcDEiBRI0VmeIMgA=</ce
c:pssh>
        </ContentProtection>
        <BaseURL>enc_tears_video.mp4</BaseURL>
        <SegmentBase indexRange="1127-1902" timescale="12288">
          <Initialization range="0-1126"/>
        </SegmentBase>
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

Upload to your streaming server

Copy your mpd and encrypted media files.
Your playback URL is the mpd file itself.

Play Content

---

1. Load the Shaka Player
    - Open a Chrome browser window to your demo player URL.
      `http://<Your Server IP Address>/shaka/`
    - Open the EME logger extension to view EME events.
    - Enter the URL to your mpd in the `Custom manifest URL` box
    - Enter the Widevine proxy server URL for `widevine_test`
      `https://widevine-proxy.appspot.com/proxy`
    - Click Load Stream

Packaging all tracks and generating the MPD at once

Once you are familiar with the tools, you can choose to encrypt all tracks and generate the MPD in one step.

For example:

```
$ packager \
input=/content/tears_audio_eng.mp4,stream=audio,output=tears_audio_eng.mp4 \
input=/content/tears_240p_600.mp4,stream=video,output=tears_240p_600.mp4 \
input=/content/tears_360p_1600.mp4,stream=video,output=tears_360p_1600.mp4 \
input=/content/tears_480p_3000.mp4,stream=video,output=tears_480p_3000.mp4 \
input=/content/tears_720p_4000.mp4,stream=video,output=tears_720p_4000.mp4 \
input=/content/tears_1080p_10000.mp4,stream=video,output=tears_1080p_10000.mp
\
--profile on-demand \
--enable_widevine_encryption \
--key_server_url
"https://license.uat.widevine.com/cenc/getcontentkey/widevine_test" \
--content_id "323031355f7465617273" \
--signer "widevine_test" \
--aes_signing_key
"1ae8ccd0e7985cc0b6203a55855a1034afc252980e970ca90e5202689f947ab9" \
--aes_signing_iv "d58ce954203b7c9a9a9d467f59839249" \
--crypto_period_duration 0 \
--mpd_output test_content.mpd
```

Using the Integration Console

This section provides an introduction to the Widevine Integration Console. The purpose of the Integration Console is to:

- Provide HTML5 playback capability of clear and encrypted content
    - Using a known-good set of sample content
    - Using your own custom content
- Allow the testing of various business rules that can be appended by the proxy
- Enable diagnostics of the license request and response payloads

The Widevine Integration Console for the Test environment (UAT) is available at **https://integration.uat.widevine.com**

Content Playback

Under Tools, click on HTML5 Player:

- The Test Player is based on the Shaka Player.
- Enter your content URL that you have generated in the lab.
- Click on the gear icon to configure your custom proxy url.
    - `https://widevine-proxy.appspot.com/proxy`
- Play your content and observe the available controls.

Having playback problems?

In Chrome browser, open the Developer Tools > Console. What does the log indicate?

Converted existing Widevine Classic content to encrypted DASH

This exercise demonstrates how to convert existing Widevine Classic (.wvm) content to DASH (clear and/or encrypted).

If a .wvm file contains multiple tracks, you will need to reference the stream ID during the conversion process. Stream IDs are listed in the parsed media information from the packager reference binary.

Example media information:

```
File "/root/sintel.wvm":
Found 4 stream(s).
Stream [0] type: Video
 codec_string: avc1.42e01f
 time_scale: 90000
 duration: 138898800 (1543.3 seconds)
 language:
 is_encrypted: true
 codec: H264
 width: 864
 height: 486
 nalu_length_size: 4

Stream [1] type: Audio
 codec_string: mp4a.40.2
 time_scale: 90000
 duration: 138898800 (1543.3 seconds)
 language:
 is_encrypted: true
 codec: AAC
 sample_bits: 16
 num_channels: 2
 sampling_frequency: 48000

Stream [2] type: Video
 codec_string: avc1.42e01f
 time_scale: 90000
 duration: 138898800 (1543.3 seconds)
 language:
 is_encrypted: true
 codec: H264
```

```
 width: 864
 height: 486
 nalu_length_size: 4

Stream [3] type: Audio
 codec_string: mp4a.40.2
 time_scale: 90000
 duration: 138898800 (1543.3 seconds)
 language:
 is_encrypted: true
 codec: AAC
 sample_bits: 16
 num_channels: 2
 sampling_frequency: 48000

Packaging completed successfully.
```

Example conversion syntax (include stream identification in the input field):

```
... input=/root/sintel.wvm,output=test.mp4,stream=0 ...
```

For simplicity, the example in this exercise includes a single track only.

1. Copy the `sintel.wvm` file

```
$ cd /root && wget https://storage.googleapis.com/wvtraining/sintel.wvm
```

    a. The file contains a single track consisting of both video and audio.
    b. It has been encrypted using `provider="widevine_test"`

2. Convert the encrypted .wvm file to un-encrypted MP4. Run the packager for conversion:
    a. Video

```
$ packager \
 input=sintel.wvm,stream=video,output=clear_sintel_video.mp4 \
--enable_widevine_decryption \
--signer "widevine_test" \
--key_server_url \
"https://license.uat.widevine.com/cenc/getcontentkey/widevine_test" \
--aes_signing_key
"1ae8ccd0e7985cc0b6203a55855a1034afc252980e970ca90e5202689f947ab9" \
--aes_signing_iv "d58ce954203b7c9a9a9d467f59839249"
```

b. Audio

```
$ packager \
 input=sintel.wvm,stream=audio,output=clear_sintel_audio.mp4 \
--enable_widevine_decryption \
--signer "widevine_test" \
--key_server_url \
 "https://license.uat.widevine.com/cenc/getcontentkey/widevine_test" \
--aes_signing_key
"1ae8ccd0e7985cc0b6203a55855a1034afc252980e970ca90e5202689f947ab9" \
--aes_signing_iv "d58ce954203b7c9a9a9d467f59839249"
```

3. Generate a new content_id in hex

```
$ echo -n <random alphanumeric string> | xxd -p
```

4. Run packager to encrypt content:

```
packager \
input=clear_sintel_video.mp4,stream=video,output=encrypted_sintel_video.mp4
\
input=clear_sintel_audio.mp4,stream=audio,output=encrypted_sintel_audio.mp4
\
--enable_widevine_encryption \
--key_server_url
"https://license.uat.widevine.com/cenc/getcontentkey/widevine_test" \
--content_id "<hex output>" \
--signer "widevine_test" \
--aes_signing_key
"1ae8ccd0e7985cc0b6203a55855a1034afc252980e970ca90e5202689f947ab9" \
--aes_signing_iv "d58ce954203b7c9a9a9d467f59839249" \
--crypto_period_duration 0 \
--mpd_output sintel.mpd
```

Note: The above syntax includes both audio and video streams to be encrypted. Additionally, it will also auto-generate the MPD file.

- Upload the files to your CDN.
- Test playback on your Shaka Player instance using the new CDN streaming URL and previously-provided proxy URL:
    - `https://widevine-proxy.appspot.com/proxy`

Packaging with multiple bitrates

1. For audio, encrypt each track separately. You will have separate encrypted audio tracks, each with their associated media_info files.
    a. Generate a new content_id in hex
        i. ***You will re-use this content_id for all the tracks in the same package.***

```
$ echo -n <random alphanumeric string> | xxd -p
```

    b.

Example:

```
$ packager
input=/root/content/128k_tears_of_steel_audio_eng.mp4,stream=audio,output=enc_tears_audio.mp4 \
--enable_widevine_encryption \
--key_server_url
"https://license.uat.widevine.com/cenc/getcontentkey/widevine_test" \
--content_id "<hex output>" \
--signer "widevine_test" \
--aes_signing_key
"1ae8ccd0e7985cc0b6203a55855a1034afc252980e970ca90e5202689f947ab9" \
--aes_signing_iv "d58ce954203b7c9a9a9d467f59839249" \
--crypto_period_duration 0 \
--output_media_info
```

2. For video, encrypt each track separately. You will have separate encrypted video tracks, each with their associated media_info files.

Example:

```
$ packager
input=/root/content/750_tears_of_steel_base_480p.mp4,stream=video,output=enc_750k_tears.mp4 \
--enable_widevine_encryption \
--key_server_url
"https://license.uat.widevine.com/cenc/getcontentkey/widevine_test" \
--content_id "<hex output>" \
--signer "widevine_test" \
--aes_signing_key
"1ae8ccd0e7985cc0b6203a55855a1034afc252980e970ca90e5202689f947ab9" \
```

```
--aes_signing_iv "d58ce954203b7c9a9a9d467f59839249" \
--crypto_period_duration 0 \
--output_media_info
```

3. Create the MPD by comma separating each input file.

```
$ mpd_generator \
--input "<include all comma separated audio and video .media_info files
created>" \
--output "large.mpd"
```

4. Upload your files to your CDN.
5. Test playback on your Shaka Player instance using the new CDN streaming URL and previously-provided proxy URL:
    a. `https://widevine-proxy.appspot.com/proxy`