Audio/Video Updates in Chrome 62

By François Beaufort
Dives into Chromium source code

- Offline playback with persistent licenses and Widevine L1 are now supported on Android.
- Chrome now disables video tracks when an MSE video is played in the background to optimize performance.
- Web developers can customize seekable range on live MSE streams.
- Chrome now supports FLAC in MP4 with MSE.
- Video will go fullscreen when the device is rotated.

Persistent licenses for Android

Persistent license in Encrypted Media Extensions (EME) means the license can be persisted on the device so that applications can load the license into memory without sending another license request to the server. This is how offline playback is supported in EME.

Until now, Chrome OS was the only platform to support persistent licenses. It is not true anymore. Playing protected content through EME while the device is offline is now possible on Android as well.

```
const config = [{
  sessionTypes: ['persistent-license'],
  videoCapabilities: [{
    contentType: 'video/webm; codecs="vp09.00.10.08"',
    robustness: 'SW_SECURE_DECODE' // Widevine L3
  }]
}];

// Chrome will prompt user if website is allowed to uniquely identify
// user's device to play protected content.
navigator.requestMediaKeySystemAccess('com.widevine.alpha', config)
.then(access => {
  // User will be able to watch encrypted content while being offline when
  // license is stored locally on device and loaded later.
})
```

```
.catch(error => {
  // Persistent licenses are not supported on this platform yet.
});
```

You can try persistent licenses yourself by checking out the Sample Media PWA and following these steps:

1. Go to https://biograf-155113.appspot.com/ttt/episode-2/
2. Click "Make available offline" and wait for the video to be downloaded.
3. Turn airplane mode on.
4. Click the "Play" button and enjoy the video!

Note: Widevine support is disabled in Incognito mode in Android. That way users do not inadvertently lose paid licenses when closing Incognito tabs.

Widevine L1 for Android

As you may already know, all Android devices are required to support Widevine Security Level 3 (Widevine L3). However there are many devices out there that also support the highest security level: Widevine Security Level 1 (Widevine L1) where all content processing, cryptography, and control is performed within the Trusted Execution Environment (TEE).

Good news! Widevine L1 is now supported in Chrome for Android so that media can be played in the most secure way. Note that it was supported already on Chrome OS.

```
const config = [{
  videoCapabilities: [{
    contentType: 'video/webm; codecs="vp09.00.10.08"',
    robustness: 'HW_SECURE_ALL' // Widevine L1
  }]
}];

// Chrome will prompt user if website is allowed to uniquely identify
// user's device to play protected content.
navigator.requestMediaKeySystemAccess('com.widevine.alpha', config)
.then(access => {
  // User will be able to watch encrypted content in the most secure way.
})
.catch(error => {
  // Widevine L1 is not supported on this platform yet.
});
```

Shaka Player, the JavaScript library for adaptive media formats (such as DASH and HLS) has a demo for you to try Widevine L1 out:

1. Go to https://shaka-player-demo.appspot.com/demo/ and click "Allow" when prompted.
2. Pick "Angel One (multicodec, multilingual, Widevine)".
3. Enter `HW_SECURE_ALL` in the "Video Robustness" field of the "Configuration" section.
4. Click the "Load" button and enjoy the video!

Background video track optimizations (MSE only)

The chrome team is always trying to find new ways to improve battery life and Chrome 62 is no exception.

Chrome now disables video tracks when the video is played in the background (e.g., in a non-visible tab) if the video uses Media Source Extensions (MSE). Check out our previous article to learn more.

Customize seekable range on live MSE streams

As you may already know, the **`seekable`** attribute contains the ranges of the media resource to which the browser can seek. Typically, it contains a single time range which starts at 0 and ends at the media resource duration. If the duration is not available though, such as a live stream, the time range may continuously change.

The good news is that you can now more effectively customize the **`seekable`** range logic with Media Source Extensions (MSE) by providing or removing a single seekable range that is union'ed with the current buffered ranges. It results in a single seekable range which fits both, when the media source duration is **`+Infinity`**.

In the code below, the media source has already been attached to a media element and contains only its init segment:

```
const mediaSource = new MediaSource();
...

mediaSource.duration = +Infinity;
// Seekable time ranges: { }
// Buffered time ranges: { }
```

```
mediaSource.setLiveSeekableRange(1 /* start */, 4 /* end */);
// Seekable time ranges: { [1.000, 4.000) }
// Buffered time ranges: { }

// Let's append a media segment that starts at 3 seconds and ends at 6.
mediaSource.sourceBuffers[0].appendBuffer(someData);
// Seekable time ranges: { [1.000, 6.000) }
// Buffered time ranges: { [3.000, 6.000) }

mediaSource.clearLiveSeekableRange();
// Seekable time ranges: { [0.000, 6.000) }
// Buffered time ranges: { [3.000, 6.000) }
```

There are many cases that I didn't cover above so I'd suggest you give a try to the official sample to see how buffered and seekable time ranges react to different MSE events.

Intent to Ship | Chromestatus Tracker | Chromium Bug

FLAC in MP4 for MSE

The lossless audio coding format FLAC has been supported in regular media playback since Chrome 56. FLAC in ISO-BMFF support (aka FLAC in MP4) was added shortly after. And now FLAC in MP4 is available in Chrome 62 for Media Source Extensions (MSE).

For info, Firefox folks are the ones who developed and implemented support for a FLAC in MP4 encapsulation spec, and the BBC has been experimenting with using that with MSE. You can read the BBC's "Delivering Radio 3 Concert Sound" post to learn more.

Here's how you can detect if FLAC in MP4 is supported for MSE:

```
if (MediaSource.isTypeSupported('audio/mp4; codecs="flac"')) {
  // TODO: Fetch data and feed it to a media source.
}
```

If you want to see a full example, check out our official sample.

Intent to Ship | Chromestatus Tracker | Chromium Bug

Automatic video goes to fullscreen when the device is rotated

If you rotate a device to landscape while a video is playing in the viewport, playback will automatically switch to fullscreen mode. Rotating the device to portrait puts the video back to windowed mode. Check out our past article for more details.

Last updated July 2, 2018.