GOING BIG SCREEN:

BRINGING VIDEO TO LG WEBOS

WRITTEN BY
PIETER-JAN SPEELMANS, CTO

LG WEBOS

“Streaming is gaining ground over traditional viewing” is something we’ve heard before. **“The big screen is the undisputed leader”** is another one of those quotes. In 2018, streaming on the big screen was however limited to about one-third of the total viewing time according to Conviva.

By 2019, this percentage had increased to 55%. So what do the latest 2020 numbers say? About 75% of all viewing happens on the big screen. When we look at the devices with the most growth, there is no way around smart TVs which grew by more than 200% during the last two consecutive years, allowing them to gain ground on other connected tv devices such as Fire TV and Roku, which grew only 55% (slightly below the 57% average across platforms).

In our previous article, we zoomed in on the largest brand in the Smart TV space, being Samsung which accounts for about 7% of all viewing time.

In this guide, we will continue with the second-largest brand: LG and its webOS platform, which accounts for half of that, ranking at 3.3%.

HOW DO WE BRING APPS TO LG'S WEBOS PLATFORM?

Since 2014, LG has been making use of webOS as a platform. While the platform has a history with Palm and HP and an open source version exists, the main driver behind webOS is LG which bought the platform in 2013. The platform itself is Linux based, but in its essence developing applications for the platform usually does not happen in languages such as C, but applications tend to run in a web environment.

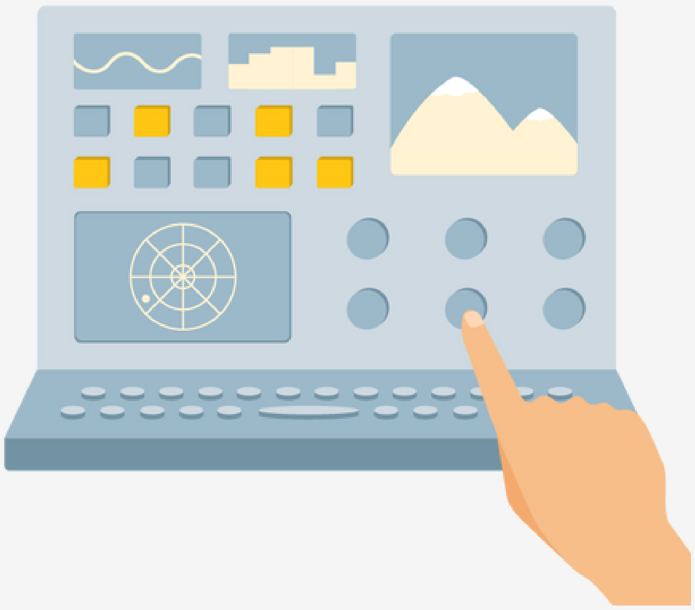
Same as most other smart TV platforms, bringing your app on LG's webOS platform goes through a store: the LG Content Store. Long story short, the LG Content Store has the same process as most application stores, requiring a (often lengthy) review before your app can be published. As in the webOS name however, apps are developed completely in web technology, being HTML5.

While the web is known for flexible updates, traditional app store processes can slow you down significantly. **On webOS however, the "hosted app" approach is quite common:** you basically create an app which instructs the platform to load the latest and greatest from the web, same as any old website. While you still have to go through the validation process once to get listed in the LG Content Store, you can bypass future reviews for updates easily giving you the agility which is needed in an era where competition in the streaming landscape is growing by the day.

Building an app for webOS (or a webOS enabled website) is quite simple thanks to web technology being the main driver. Most APIs which are used are following the W3C specifications nicely, making it **highly similar to developing your website for any common browser**. Actually, you can likely reuse most of your website logic, given some tweaking on the UX side to ensure smooth integrations with LG's Magic Remote. While the Magic Remote allows you to summon a mouse-like cursor on your screen to navigate (which acts from an API perspective in the same way as any odd mouse on a laptop), using standard remote control buttons (which from an API perspective function as a keyboard) still works quite well (and in most apps, better).

In order to bring your content to webOS, you of course need to play it. On webOS this is quite simple. You can bring audio or video streaming to webOS in exactly the same way you do on your website:

1. You either rely on the native player for basic playback
2. Or you load up a "bring your own" video player library to power your unique viewer experiences.



IS NATIVE VS "BRING YOUR OWN" REALLY A CHOICE TO MAKE?

If you would ask me, the answer is quite simply… no. **The native playback support on webOS is extremely rudimentary**. While it can be started in exactly the same way as you do on any odd browser in a video-tag as simple as:

```
<video src ="https://cdn.theoplayer.com/videos/bbb.m3u8"></video>
```

You will be extremely limited when using this approach. While adding in DRM is still possible, reading through the webOS "Supported Media and DRM Formats"-page reads like a horror-story for the average media publisher. **The most important limitations?**

1. No real support for MPEG-DASH. Officially, even basic support is lacking. Where most content publishers are moving to a setup where MPEG-DASH is combined with Common Encryption with Widevine and PlayReady, webOS got stuck in the HLS era. In practice, there is basic support for MPEG-DASH: VOD feeds are partially supported, but live (and especially timeshift) support tends to be very shaky.

2. Officially, the use of Smooth Streaming is discouraged. While support is available for both VOD and Live (with a number of limitations, but basics such as timeshift are available), it is not recommended to be used. While it is a great path for people to take when wanting to use a more standardised set of streams, it's not really an option if you want official LG support.

3 Even HLS support is basic. Most modern webOS 5 TVs support HLS version 7 (which is Pantos-draft 14, straight out of 2014). The 4.x version of the platform only supports HLS version 5, and if you're unlucky, you are stuck on HLS version 3 for all older devices.

4 Even with HLS support, the list of unsupported HLS tags seems to be longer than the list of supported tags. There are a large amount of limitations when reading through the availability of EXT-X-DISCONTINUITY (which is crucial for SSAI) and metadata doesn't seem to have support at all as neither EXT-X-PROGRAM-DATE-TIME nor EXT-X-DATERANGE (ID3 isn't handled either).

5 You're stuck with WebVTT. It's the only subtitle format even listed… That hurts.

6 At least there is DRM: PlayReady and Widevine Modular work… if your device is webOS 3.0+. For older versions, it's down to Widevine Classic, but don't worry, that's supported on most newer devices as well! On a side-note: PlayReady support is available on older models of webOS as well, but again, it's undocumented and officially, it's not there…

NOTABLE PROPERTIES

MODEL VERSION

2014 Models
webOS 1.x

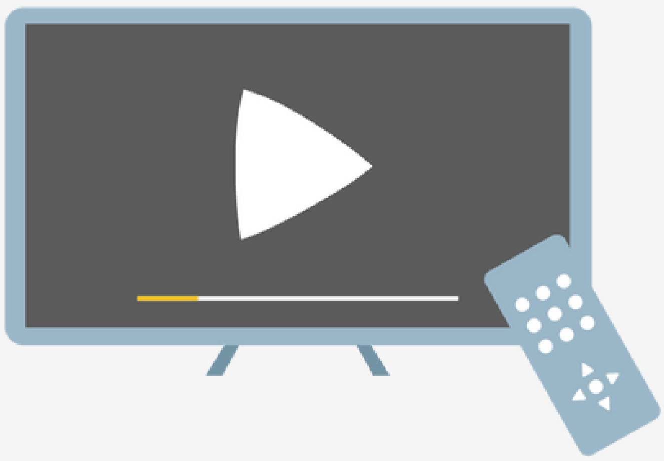
- WebKit 537.41 (but the browser uses Chromium).
- HLS version 3 (Pantos Draft 5 November 2011).
    - No EXT-X-MEDIA which is needed for alternative audio and subtitle tracks.
    - MPEG-TS or raw audio only (no CMAF).
    - webOS limitations:
        - EXT-X-DISCONTINUITY is only supported for timestamp. modification, not for codecs etc.
        - EXT-X-PROGRAM-DATE-TIME is not supported.
        - EXT-X-ALLOW-CACHE is not supported.
- Fast forward & reverse are not supported.
- No live seeking (time-shift mode),
- WebVTT subtitles only (side-loaded).

MODEL VERSION	NOTABLE PROPERTIES
2014 Models webOS 1.x continued...	- DRM: - Widevine Classic version 6. - AES-128 clear key encryption. - Must be played using the DRM Service, no support for EME (see later). - Only 1 DRM enabled feed can be loaded in parallel. - Unofficially supports Smooth Streaming and (basic) MPEG-DASH. For MPEG-DASH, live streaming and time shifting is only partially supported. When attempting to do time shifting, the playback can crash completely. PlayReady 2.0 and Verimatrix 1.0.7 is supported unofficially on the DRM front as well.
2015 Models webOS 2.x	- Identical to webOS 1.x except for the WebKit engine being updated to 538.2 (the browser still uses Chromium). - Unofficial support for PlayReady is at version 2.5, Verimatrix at 3.4.0.
2016-2017 Models webOS 3.x	- Chromium 38. - HLS version 3 (Pantos Draft 5 November 2011). - See webOS 1.x for limitations. - Fast forward & reverse are not supported. - Live seeking supported (time-shift mode). - WebVTT subtitles only (side-loaded). - DRM: - Widevine Classic version 6. - AES-128 clear key encryption. - PlayReady support. - Widevine Modular version 2.2 or 3.0.5 (for webOS 3.5) supported. - Only 1 DRM enabled feed can be loaded in parallel. - Unofficially supports Smooth Streaming and (basic) MPEG-DASH. For MPEG-DASH, support is better compared to older versions. Unofficial PlayReady and Verimatrix support are at 2.5 and 3.4.0 respectively.

MODEL VERSION	NOTABLE PROPERTIES
2018-2019 Models webOS 4.x	- Chromium 53. - HLS version 5 (Pantos Draft 9 September 2012). - MPEG-TS or raw audio only (no CMAF). - webOS limitations: - EXT-X-DISCONTINUITY is only supported for timestamp modification, not for codecs etc. - EXT-X-PROGRAM-DATE-TIME is not supported. - EXT-X-ALLOW-CACHE is not supported. - EXT-X-I-FRAMES-ONLY, EXT-X-I-FRAME-STREAM-INF are not supported. - EXT-X-BYTERANGE is only “partially supported”. - EXT-X-MEDIA does not seem to be supported at all (no multi-audio/video/subtitles). - Fast forward & reverse are not supported. - Live seeking supported (time-shift mode). - WebVTT subtitles only (side-loaded). - DRM: - Widevine Classic version 6. - AES-128 clear key encryption. - PlayReady 3.0 support. - Widevine Modular version 3.2.1 or 4 (for webOS 5.5) supported. - Unofficial Smooth Streaming, MPEG-DASH and DRM support is at the same level as 3.x versions.
2020 Models webOS 5.x	- Chromium 68. - HLS version 7 (Pantos Draft 14 October 2014). - MPEG-TS or raw audio only (no CMAF). - webOS limitations: - EXT-X-DISCONTINUITY is only supported for timestamp modification, not for codecs etc. - EXT-X-PROGRAM-DATE-TIME is not supported. - EXT-X-ALLOW-CACHE is not supported. - EXT-X-MEDIA with multi-audio is only “partially supported”, for webVTT subtitles the same applies.

MODEL VERSION	NOTABLE PROPERTIES
2020 Models webOS 5.x	- Fast forward & reverse are not supported. - Live seeking supported (time-shift mode). - WebVTT subtitles only (side-loaded). - DRM: - Widevine Classic version 6. - AES-128 clear key encryption. - PlayReady 4.0 support. - Widevine Modular version 15 supported. - Unofficial Smooth Streaming, MPEG-DASH and DRM support is at the same level as 3.x versions.

So, as you can see, the limitations of the native player are quite severe. The fact there is no MPEG-DASH support, no real support for SSAI (due to a lack of support for a full EXT-X-DISCONTINUITY) and only the less common support for Widevine and PlayReady in HLS, the native player is severely handicapped. There is however some light at the end of the webOS "Supported Media and DRM Formats"-page: MSE and EME support.



LEVERAGING MSE AND EME TO BRING YOUR OWN VIDEO ON WEBOS

The Media Source Extension (MSE) API and the Encrypted Media Extension (EME) API are available on all later webOS models. Where MSE provides access to media decoders, EME allows you to access the Content Decryption Module (or CDM) which allows you to leverage the platform's DRM capabilities.

Using **the MSE/EME approach is far from a new thing when it comes to media playback**. Native players on most browser platforms are often even more limited compared to webOS' native media playback capabilities. As a result **nearly every website playing media is making use of these APIs to power high end media experiences**. If you have a website, odds are pretty high you are doing the same thing (unless you are still stuck on Flash for some reason).

Where THEOplayer was the first media player to do complex handling of media in browser environments (even without the use of MSE/EME), various other companies started providing libraries since the APIs became available. As MSE and EME are only APIs, there is however more than meets the eye: **development based on these APIs still requires a proper understanding of media and the web as the entire implementation of the streaming protocols has to happen in dedicated code.** In most cases, fully featured media players have to go above and beyond that in order to bring a truly rich viewer experience.

Even with all of the basics implemented, there is a large difference between media player options. While some media players only provide basic playback such as open-source libraries hls.js, dash.js and Shaka Player providing support for the HLS or MPEG-DASH protocol, **more advanced players such as THEOplayer provide even more support such as integrations with various DRM systems, monetisation options with CSAI and SSAI, picture in picture support, support for offline downloading, ...**

While webOS does bring support for all of the required APIs to use any framework, there are some significant differences in implementations.

On one end, most (open source) playback implementations assume the MSE and EME APIs are fully up to date. On webOS however this is not the case as you can see in the table below:

YEAR	VERSION	BROWSER	MSE	EME
2014	webOS 1.x	538.2 (Sept 2014)	N/A	N/A
2015	webOS 2.x	538.2 (Sept 2014)	N/A	N/A
2016	webOS 3.x	M38 (Oct 2014)	Dec 2013 Draft	July 2012 0.1b
2017	webOS 3.x	M38 (Oct 2014)	Dec 2013 Draft	July 2012 0.1b
2018	webOS 4.x	M53 (Aug 2016)	Oct 2016 Draft	Mar 2015 Draft
2019	webOS 4.x	M53 (Aug 2016)	Oct 2016 Draft	Mar 2015 Draft
2020	webOS 5.x	M68 (July 2018)	Nov 2016 Rec.	Sept 2017 Rec.

As MSE/EME implementations can be quite old on webOS (with only webOS 5.x to be on the recommendation), it's crucial to have a playback library which works with those old versions. On top of this, webOS also tends to have specific decoder behavior due to hardware being significantly different from standard browser environments on laptop or mobile. This can be expressed in things such as seeking only to key frames on specific versions, not allowing changes in playback rate, how the buffer is exposed etc. All of this means **it is crucial your playback library supports all webOS versions where you plan to deploy**.

While they usually work on the platform, **most common open source playback rarely have official support for webOS**. With smart TV platforms being forgotten and not making it into the testing cycle this opens up a risk. For example, the compatibility list of hls.js does not list webOS, dash.js doesn't list any platforms at all and only claims to rely on MSE being implemented and Shaka Player mentions webOS is not officially supported, but "expected to work". **At THEOplayer, we actively support the webOS platform** and have code to handle differences and quirks compared to a standard browser environments in our THEOplayer Web SDK such as handling for limitations in the number of decoders which can be used in parallel, switching from one video codec into another, enabling and disabling DRM, handling for higher bitrates, alternative codecs and HDR content on big screens just to name a few.

The compatibility problem doesn't just show in regard to testing. As already mentioned earlier, there are still significant differences in the web environment, MSE and EME implementations across different Tizen versions. **Where most players assume an always up-to-date API, this is not the case on smart TV platforms such as webOS for which different models often support draft specifications only.**

WHERE TO GO FROM HERE?

As discussed in this Guide, **the best approach for webOS is to leverage MSE/EME where available**. The limitations of the native pipeline are too significant and block you from delivering a high-quality user experience.

There are however some platforms where this is not possible, being webOS 2.x and webOS 1.x. For those platforms, you will be condemned to use the native playback pipeline. As a result, it makes sense to carefully evaluate if you want to support webOS 1.x and 2.x. If the usage footprint for your audience is not large enough, you might be incurring more costs than you want as you will probably need to set up custom streams using HLS with PlayReady and will lose a number of capabilities such as time shifting, SSAI monetisation and multiple audio. Even the basics such as subtitles will become hard.

With THEOplayer, we do aim to make this go down as smoothly as possible, providing you the ability to switch to native playback if you desire this, while still making use of our universal API. This has the added benefit integrations with external services such as analytics and DRM only have to be implemented once. It effectively allows you to use a hybrid approach: making use of native playback where it makes sense and leveraging the power of MSE/EME where it is possible.

The most important advice? Weigh your options. Making a wrong choice can set you back significantly if you have to switch at a later point in time. Any questions left? Don’t hesitate to reach out to our team.

THEO

INTERESTED IN BRINGING VIDEO TO LG'S WEBOS?

Get in contact with one of our THEO experts.

PIETER-JAN SPEELMANS
CTO & Founder

WILLEM DE SAEGHER
VP Global Sales

Ask an expert