THEO

GOING BIG SCREEN

BRINGING VIDEO TO SAMSUNG TIZEN

WRITTEN BY
PIETER-JAN SPEELMANS, CTO

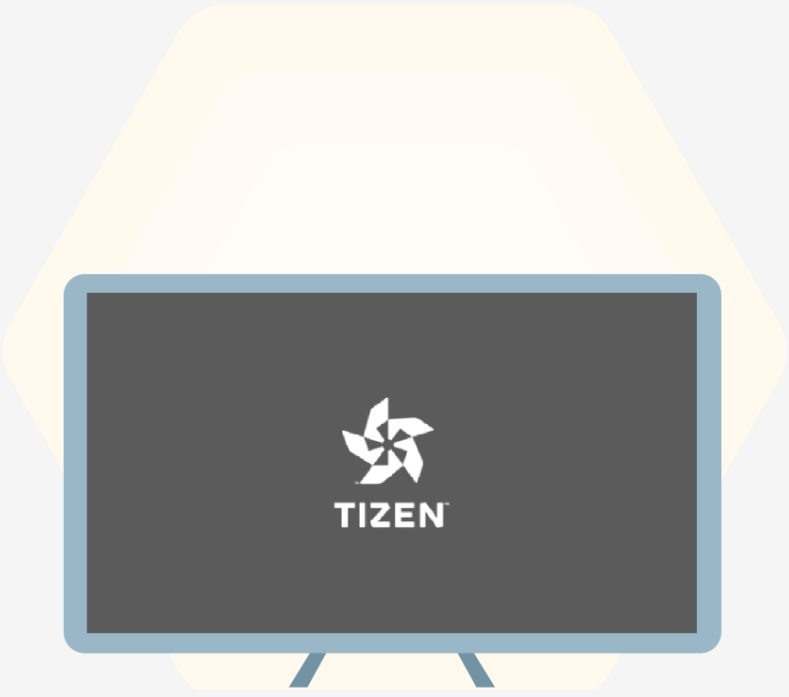
SAMSUNG TIZEN OS

As recent as the last few years, online streaming has evolved tremendously. Initially, most of the streaming happened on desktop devices, but today that is not the case. Today, video streaming is conquering the big screen (Smart TVs).

Without a doubt, one of the **most notable big screens is Samsung with the Tizen operating system**. Based on Conviva's State of Streaming from 2020 Q4, **Samsung accounts for roughly 49% of streaming on Smart TV**, leaving other vendors such as **LG (±22%)** and **Vizio (±12%)** in the rearview. Due to the growing popularity, and almost **50M smart Samsung Smart TVs being shipped every year**, interest to deploy OTT apps directly on Samsung Smart TVs has risen significantly.

While deploying an OTT app on the Tizen operating system is far from complex, there are a number of caveats.

In this guide, we will go through the **possible approaches to deliver apps and getting video playback,** as well as shed some light on **how to tackle the large set of different devices** and the problem with different Tizen versions.



DEVELOPING APPS ON TIZEN

When your goal is to bring streaming to the big screen, Samsung's Tizen is a great place to start. Its popularity and fast evolution over the past years make it a solid choice for an investment in your streaming apps' platform expansion. Tizen apps can be installed from the Samsung Apps TV store.

Publishing an app in this app store isn't all too difficult. Same as with most app stores for other platforms, **applications are to be submitted for review to Samsung and will, once approved, be published in the store.** The disadvantage to this process is that you need to go through it again for any update you want to perform to your application.

When talking about video apps, one of the most crucial choices to make is the choice for a video player as this will drive a big part of the viewer experience, as well as the number of features your own team will need to build and maintain themselves.

In general, there are two approaches to choose from:

1. Leveraging the streaming support of the native video player AVPlay.
2. Use a "Build Your Own" video player approach leveraging available advanced APIs.

WHY DEVELOP A HOSTED APP FOR TIZEN?

When developing an application for Tizen, it's important to get into the Samsung Apps TV-store. This requires your app to get reviewed. As reviews can take a long time, Hosted Apps are often leveraged. A hosted app is a thin wrapper app that loads all assets externally from a web server instead of bundling them into the app (a so-called Packaged App).

By caching these assets and leveraging and techniques similar to Progressive Web Apps (PWAs), users still get instant access to the application while having the advantage of fast and easy updates without going through the app store review. The Hosted App approach also allows for A/B testing where different users get to use slightly different variants of the app. This allows making informed decisions to be made regarding the impact on viewer behavior before a mass roll-out.

APPROACH 1

GOING NATIVE: LEVERAGING AV PLAY

As with most smart TV systems, Tizen comes with its own native video player, providing support for the most common streaming protocols. Samsung Tizen's native video player is called AVPlay. Using AVPlay is not too difficult, and simply requires loading the AVPlay API library. This is as simple as loading the following script:

```
<SCRIPT TYPE="TEXT/JAVASCRIPT"SRC="$WEBAPIS/WEBAPIS/WEBAPIS.JS" </SCRIPT>
```

With this library loaded, you can enable playback easily with AVPlay API by:

1. Defining the playback window and indicating AVPlay is to be used,

```
var objElem = document.createElement('object');
objElem.type = 'application/avplayer';
document.body.appendChild(objElem);
```

2. Defining the playback window and indicating AVPlay is to be used,

```
webapis.avplay.open('https://content.domain.com/channel1.m3u8');
```

4. Loading the media source to prepare for playback,

```
webapis.avplay.prepare();
```

3. and trigger playback.

```
webapis.avplay.play();
```

As you can see, getting playback up and running is very simple. Beyond this, AVPlay brings some additional capabilities such as:

- Support for streaming protocols such as **Smooth Streaming, MPEG-DASH, and HTTP Live Streaming (HLS)**,
- Allowing to perform **basic trick play** by seeking or changing the playback speed.,

- The ability to receive notifications in case of buffering or errors,
- Support for alternative audio channels,
- Basic DRM playback, and
- Support for subtitle formats such as SMPTE-TT, SAMI & DFXP.

AVPlay has been around since Tizen's early days in 2015 (with the launch of Tizen 2.3) and has evolved a lot, but as a result, **AVPlay differs significantly in capabilities across different platform versions of Tizen**.

The main purpose of AVPlay however hasn't changed: empower media apps to use basic playback capabilities. **However, when building a full media app, in practice, the capabilities of AVPlay will soon reach their limitations**.

There are two main challenges which OTT video publishers often start to encounter when working with AVPlay:

1. Capabilities are limited to basic use cases and offer little flexibility,
2. Different versions of AVPlay are tied down to the support of specific versions of streaming protocols and (greatly) differ in capabilities.

AVplay's Limitations

Where the limited basic use cases support is often initially not seen as a huge issue, the problems start growing once new business cases get added. Some notable use cases which are **difficult or even impossible to support properly are:**

- Monetization through client-side (CSAI) or server-side ad insertion (SSAI).
- Reducing playback latency for optimal user experience in interactive or live broadcasts.
- Advanced bitrate selection (or even manual bitrate selection).
- Identifying streaming issues as error responses are extremely brief.
- Monitoring QoE due to lack of detailed events (no possibility to inspect the size of the buffer, details of different tracks and qualities, …).
- Support for alternative subtitle formats and subtitle styling.
- Improved scrubbing with thumbnails or I-FRAME streams.

On top of these limitations, **the support for streaming protocols differs significantly between different versions of AVPlay**. The root cause of this seems to be the **lack of software updates for Samsung TVs**: Samsung tends to have a policy of locking models released in specific years to the Tizen version released that year. Samsung TVs from 2015 will be locked to Tizen 2.3, the 2016 version to 2.4, 2017 to 3.0, and so forth. At the time of writing, the 2020 models are running Tizen 5.5.

As a result, if you are an OTT video publisher and you want to support different models of Samsung Smart TVs, some even only a couple of years old, AVPlay will cause some headaches for your product and engineering teams. For example, a stream tested on a 2017 model Tizen 3.0 device with alternative audio tracks will not work properly on the 2016 model with Tizen 2.4.

> ***If you are an OTT publisher and you want to support different models of Samsung Smart TVs, some even only a couple of years old, AVPlay will cause some headaches for your product and engineering teams.***

These and similar limitations force you to either set up **a specific stream for every Tizen version** and incrementally add capabilities and features, or go for the **lowest denominator approach** which means limited capabilities across all devices. This decision can have a huge impact on both operational cost as well as viewer experience.

To make things even worse, while there is a relatively clear overview available on HLS protocol version support, this is not the case for MPEG-DASH support and the different IOPs which were published. A list of the most important differences in AVPlay's support between recent Tizen versions can be found on the next following pages. Note that additional limitations on top of the ones listed below are in effect as well.

MODEL VERSION	NOTABLE PROPERTIES	
2015 Models Tizen 2.3	- DRM: - PlayReady 2.5 with SL2000 (software-CENC/CTR only) - Widevine Classic only (no Modular) - HLS: - No support for session keys - No byte-range request support - All limitations of more recent versions	- MPEG-DASH: - All limitations of more recent versions
2016 Models Tizen 2.4	- DRM: - PlayReady 2.5 with SL2000 (software – CENC/CTR only) - Widevine Modular 2.08 - HLS (limited version 7 support): - No support for discontinuity sequence (improper handling of SSAI) - No support for alternative audio tracks - No support for in-band subtitles (WebVTT) - No support for in-band captions (CEA-608) - All limitations of more recent versions	- MPEG-DASH: - Not following DASH interop on HEVC usage - No support for alternative audio tracks - No support for in-band subtitles - No support for trick modes - No support for in-band captions (CEA-608) - All limitations of more recent versions
2017 Models Tizen 3.0	-DRM: - PlayReady 2.5 with SL2000 (software – CENC/CTR only) - Widevine Modular 2.08 - HLS (limited version 7 support): - All limitations of more recent versions	- MPEG-DASH: - No support for key rotation - All limitations of more recent versions

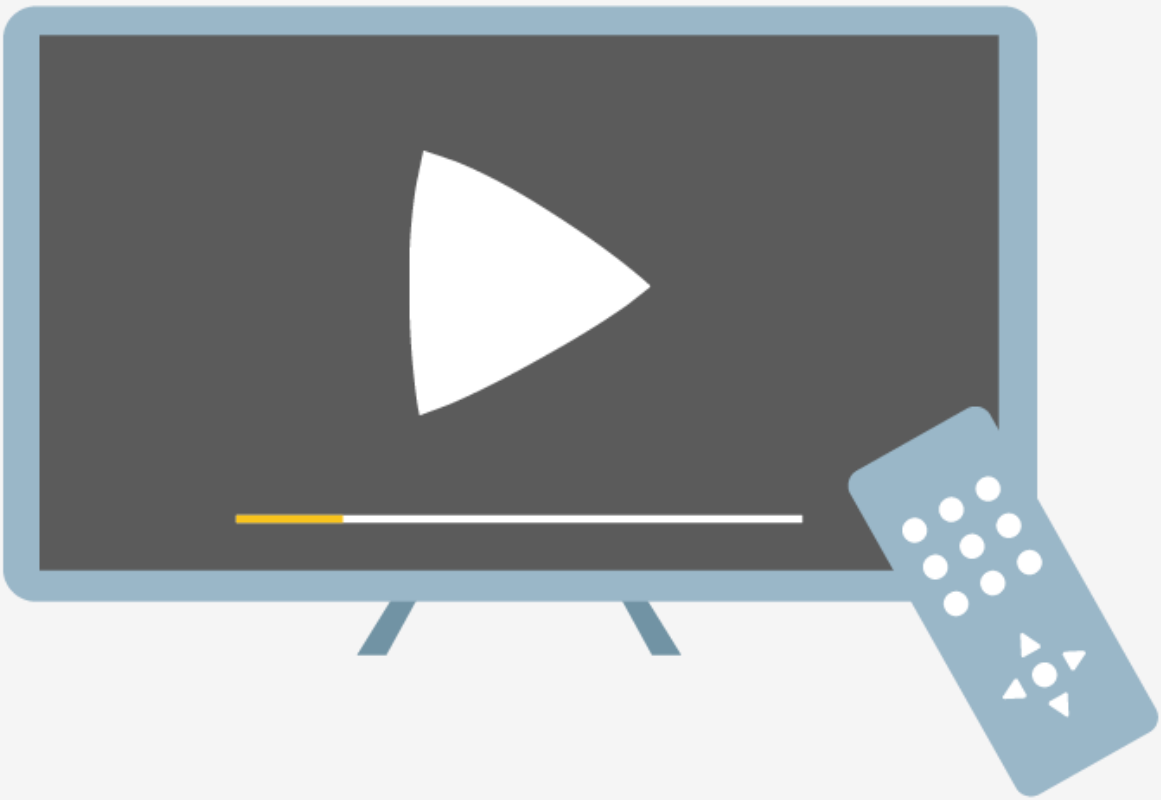
2018 Models
Tizen 4.0
	- DRM:
	  - PlayReady 3.3 with SL3000 (hardware – CENC/CTR only)
	  - Widevine Modular 3.2
	- HLS (limited version 7 support):
	  - All limitations of more recent versions
		- MPEG-DASH:
		  - All limitations of more recent versions

2019 Models
Tizen 5.0
	- DRM:
	  - PlayReady 3.3 with SL3000 (hardware – CENC/CTR only)
	  - Widevine Modular 14.1
	- HLS (limited version 7 support):
	  - All limitations of more recent versions
		- MPEG-DASH:
		  - All limitations of more recent versions

2020 Models
Tizen 5.5
	-DRM:
	  - PlayReady 4.2 with SL3000 (hardware, including CBCS support)
	  - Widevine Modular 15.2
	- HLS (limited version 7 support):
	  - No low latency support
	  - No support for CMAF or HEVC
	  - No metadata support (daterange, program date time, …)
	  - No support for initializations through MAP
	  - No support for start offsets
	  - No support for IMSC1 subtitles
	  - No support for alternative video tracks
	  - No support for parts, delta playlists, gap marking, preload hints, …
		- MPEG-DASH:
		  - No low latency support
		  - No SSAI: multi period support is not stable
		  - Mixed results for enabling/disabling DRM and rotating keys
		  - No metadata support (eventstream, emsg, …)
		  - No support for alternative video tracks

As a result of these limitations, the **AVPlay APIs provide a very simple approach for OTT video publishers to get up and running fast** with basic OTT video apps, **but once video requirements grow, this approach quickly shows its limitations.**

Furthermore, if it is your ambition to target Tizen versions as of 2015 or 2016, it is often needed to have specific streams set up in order to be compatible with Tizen. This will result in duplicating pipelines and multiplying stream generation and operational distribution costs.

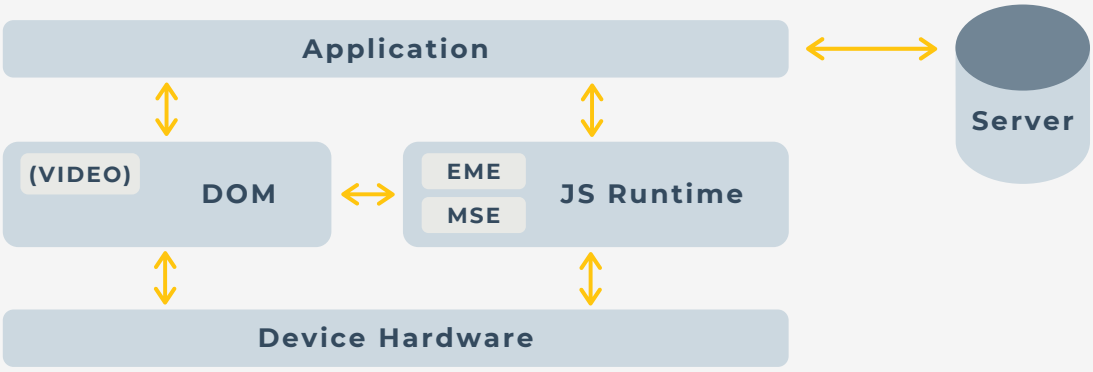


APPROACH 2

BRING YOUR OWN: MSE/EME

Due to the limitations of native players such as AVPlay, **the popularity of the "use your own" approach to video players is growing rapidly** with OTT video publishers. This approach allows avoiding the biggest issues with AVPlay due to the differences between Samsung TV models and Tizen versions. **When taking the bring your own player approach, it is important that hardware capabilities for DRM decryption and media decoding can be leveraged.** This has an impact on viewer experience as it allows to unlock capabilities such as HDR but also is often important for battery efficiency.

On Samsung's Tizen platform media hardware access is possible thanks to the presence of the Media Source Extension (MSE) API and the Encrypted Media Extension (EME) API. Tizen has support for different versions of these APIs since Tizen 2.3 (2015 models). While versions differ, support is broad enough to implement all the most important use cases with only a limited set of limitations (more on this later).



The MSE API provides access to the decoders. Samsung has gradually extended support on this over the last years, increasing frame rates, resolutions, and bit-rates as devices became more powerful. The API allows an application to allocate buffers on the audio and video decoders of the device and feed it with media data. This allows the implementation of common streaming protocols such as HLS and MPEG-DASH (as well as others). A video player can download the required playlist or manifest, identify the relevant segments, download them and append them one after the other as playback happens.

Leveraging MSE for basic playback can be quite simple. In a rudimentary form, you should use the following steps:

1. Get a handle on an HTML5 Video Element (or `<video>` tag).

```
var video =
document.getElementById('video');
```

2. Create a new `MediaSource` through the MSE-API.

```
var mediaSource = new mediaSource();
```

3. Wait for the `MediaSource` to "open" to create a `SourceBuffer` (the direct interface to the decoder's buffer where you can store raw data) and load it in the video element.

```
mediaSource.addEventListener('sourceopen',function(e){
    // ... var videoSourceBuffer =
mediaSource.addSourceBuffer('videoMimeType');
    // ... }); video.src =
window.URL.createObjectURL(mediaSource);
```

4. Download and parse the right playlist for your stream (keeping in mind bitrate selection) and identify the segments you want to load.

5. Download the appropriate segments and append them to your `SourceBuffer`.

```
while (nextSegment) {var segment = await
download(nextSegment.uri);sourceBuffer.appendBuffer
(segment);nextSegment = findNextSegment(playlist);}
```

In parallel, **the EME API enables applications to playback DRM-protected content** and provides access to the Content Decryption Modules (CDMs) which are available on the platform.

On Tizen this allows you to access the underlying key systems for PlayReady & Widevine. When media is protected with one of these DRMs, the EME API will expose this and allow the application to configure the CDM with a response to its license challenge. Leveraging the Encrypted Media Extensions API to enable playback of DRM protected content is relatively straightforward:

1. Request access to the `KeySystem`.

```
navigator.request MediaKeySystemAccess
('com.widevine.alpha',options);
```

2. Create a `MediaKeys` object on which the media session will live later on.

```
var mediaKeys = keySystemAccess.createMediaKeys();
```

3. Listen for an `encrypted`-event indicating a new DRM session to be started.

```
video. addEventListener('encrypted',
function (event) {
    var keySession = mediaKeys.createSession();
keySession.generateRequest(event.initDataType,
event.initData);
    // ...
});
```

4. Handle messages from the session when it faces encrypted content with a key it doesn't know.

```
keySession.addEventListener("message",
function (event) {
 var request = event.message;
 var response = await
postRequest(licenseServerURI, request);
keySession.update(response);
});
```

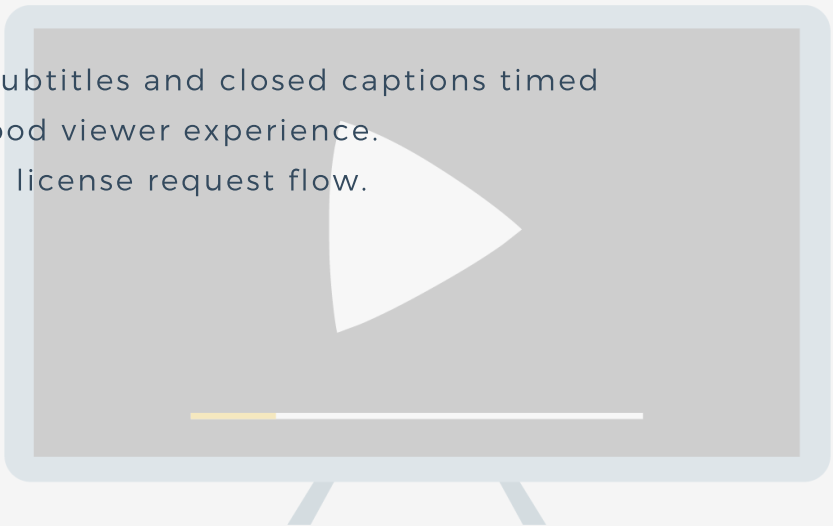
5. With the license response, the CDN can then start the decryption of the content so it can be played.

Thanks to these MSE/EME APIs, developing video playback with complex handling of media became possible. Where THEOplayer was the first media player to do complex handling of streaming media in browser environments (even without the use of MSE/EME), various other companies started providing libraries since the APIs become 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.**

The very simple examples earlier in this article can already provide basic playback of the downloaded segments. A fully-featured media player however does much more including:

- Identifying user behavior such as playing/pausing.
- Reacting to the user seeking to a new timestamp.
- Modifying the media player's volume.
- Following the appropriate autoplay policies to start playback upon the application being opened.
- Providing support for changing the playback speed.
- Monitoring the available network environment and dynamically switching to the appropriate bitrate.
- Ensure continued playback in case media data is unavailable.
- Avoiding and handling buffer underrun (and the resulting stalling behavior for viewers).
- Exposing alternative audio and video tracks to the viewer and allow them to switch.
- Retrieving and overlaying subtitles and closed captions timed exactly right to ensure a good viewer experience.
- Handling errors in the DRM license request flow.
- And much, much more.



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

The most important parameter to take into account when picking a media player for Tizen is the list of platforms and versions which are supported. **Most popular libraries are built with web browsers on laptops and mobile in mind**. Often the assumption is made all browsers implement all APIs in exactly the same way. Browser implementations can however differ significantly. Also, the Tizen platform and smart TVs in general often have significantly different hardware and use cases compared to standard playback on laptops.

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 Tizen, dash.js doesn't list any platforms at all and only claims to rely on MSE being implemented and Shaka Player only mentions Tizen 2017 models as actively tested and supported.

At THEOplayer, we actively support the Tizen platform and have code to handle differences and quirks compared to a standard browser environment 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 Tizen for which different models often support older draft specifications only.**

For example, Tizen 3.0 (models from 2017) makes use of Chromium M47, a version from December 2015 and boasts a November 2015 based MSE implementation and an EME version from February 2016. This is far from the recent specification and can have a significant impact on compatibility.

An overview of differences in MSE and EME support for different Tizen models and versions can be found below:

MODEL VERSION	WEB ENGINE	MSE VERSION	EME VERSION	NOTABLE QUIRKS
2015 Models Tizen 2.3	Webkit r152340	Dec 2013 (Draft)	Aug 2012 (Draft 0.1b)	Hard to implement DRM ("Promise-bug" in EME) DRM toggling limitations (SSAI) Codec switching limitations (SSAI) Only seeking to keyframes
2016 Models Tizen 2.4	Webkit r152340	Dec 2013 (Draft)	Aug 2012 (Draft 0.1b)	Same as 2015 Models
2017 Models Tizen 3.0	Chromium M47 (Dec 2015)	Nov 2015 (Draft)	Feb 2016 (Draft)	DRM toggling limitations (SSAI) Codec switching limitations (SSAI) Only seeking to keyframes allowed
2018 Models Tizen 4.0	Chromium M56 (Jan 2017)	July 2016 (Draft)	July 2016 (Draft)	Codec switching limitations (SSAI)
2019 Models Tizen 5.0	Chromium M63 (Sept 2017)	Nov 2016 (Rec.)	Sept 2017 (Rec.)	Mostly OK
2020 Models Tizen 5.5	Chromium M69 (Sept 2018)	Nov 2016 (Rec.)	Sept 2017 (Rec.)	Mostly OK

BRINGING VIDEO TO TIZEN: KEY TAKEAWAYS

1. Bringing video efficiently to all different versions of Tizen is a challenge, but your viewers are across a wide range of different Tizen versions, therefore **premium OTT video service can mostly not simply ignore these older platforms.**

2. Native playback is available using **AVPlay**, but it comes at a cost. Where it is often tempting to go with Tizen's native AVPlay streaming support as it is easy to get started. It is in practice often **too limiting for premium OTT video providers and can incur issues due to different support across versions.** This becomes especially apparent you're your use case includes DRM or SSAI.

> ***Native playback might help you start quickly and easily, but its limitations are too significant, especially if you require DRM or SSAI support.***

3. You can use a third party or your own video player on Tizen. MSE and EME make this possible. As there are still a broad variety of different versions of the APIs available, it is crucial that the video player you use is optimized as well for the older versions used in Tizen. Due to the quirks in the Tizen platform and its decoders, it is also crucial to ensure your player optimizes for these in order to avoid hiccups during playback.

   When picking which player to use, make sure to check compatibility: **try to avoid making the choice for a player which does not really support the Tizen versions** you want to support. Unless if you are willing to pick up the testing, bug fixing (were even possible) and maintenance internally.

4. You could use a Hybrid Approach where you leverage a “Use Your Own Video Player” approach, and fall back to AVPlay for other use cases. THEOplayer allows configuring on which version you want to use the MSE/EME approach, and on which versions you want to leverage AVPlay through a single API for configuration and interaction with other systems such as QoE and QoS monitoring. In practice, we see it is most interesting to leverage MSE/EME, especially for later versions of Tizen.

> ***Hybrid Approach allows you to have the best of both worlds: extensive features of a more capable player while falling back on AVPlay for difficult-to-support versions.***

For really early versions of Tizen, using a Hybrid Approach allows you to enable AVPlay on those specific old versions where it still makes sense in your target audience. The Hybrid Approach can bring the best of both worlds with extensive capabilities using a UYO video player while falling back on AVPlay for difficult to support versions. Choosing your video player approach is subject to careful decision-making to ensure all key features are present and compatibility across different versions of Tizen is guaranteed for your use cases. **Making the wrong choice in this strategy can cause a significant setback** if at a later point in time the player has to be swapped out for a different one.

INTERESTED IN BRINGING YOUR VIDEO TO SAMSUNG TIZEN?

Get in contact with one of our THEO experts.

PIETER-JAN SPEELMANS
CTO & Founder

WILLEM DE SAEGHER
VP Global Sales

Ask an expert