

WHITEPAPER

DIY with Open Source vs. Commercial Video Player Approach

Navigating Cross-Platform Video
Playback Complexities

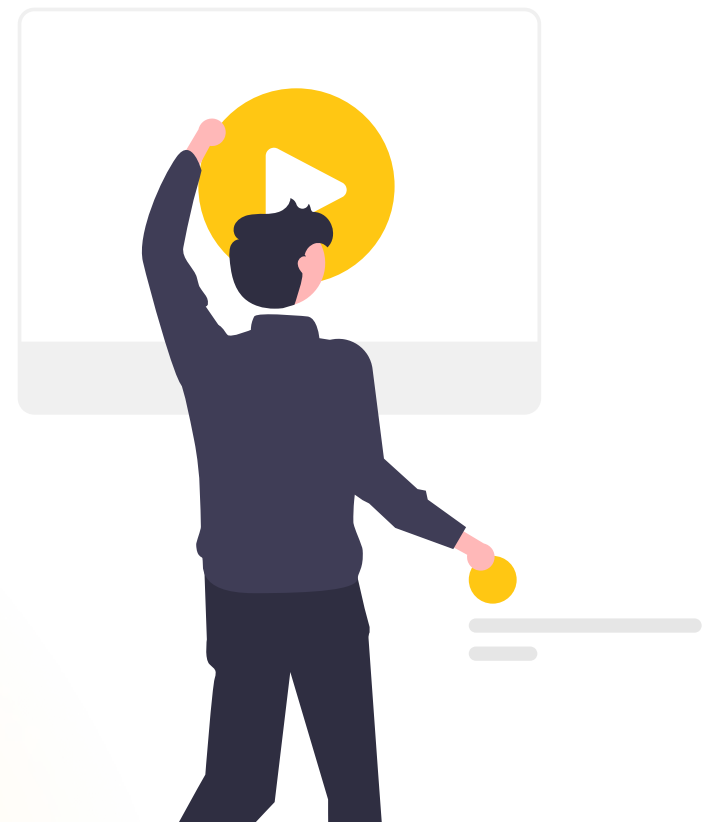


TABLE OF CONTENTS

03

INTRODUCTION

Cross-platform video playback is table stakes.....	03
The importance of a video player.....	04
Video player evaluation criteria.....	06

07

DIY with open source vs. commercial video player approach

10

Pros and cons of both approaches

Time-to-market.....	12
Reliability & testing.....	15
Technical support levels.....	17
Maintenance effort & total cost of ownership.....	19

22

Conclusion

23

Annex 1: Checklist

24

Annex 2: Overview of open source and native video players

AVPlayer.....	24
dash.js.....	24
ExoPlayer.....	25
hls.js.....	25
Shaka Player.....	26
Roku Video Node.....	26
video.js.....	27

Introduction

Cross-platform video playback is table stakes

Online video streaming is at the fingertips of users, available anytime, anywhere, and on any device. Major streaming services such as Netflix, YouTube, Peacock, Twitch, and others have set industry benchmarks, shaping user expectations for platform accessibility. Viewers have grown accustomed to the seamless streaming experiences offered by these giants, irrespective of their chosen devices.

To capture and retain user attention, streaming services must be available across a spectrum of devices. Most Over-The-Top (OTT) video services currently support **over 10 platforms**, ranging from Web, Android, and iOS to big-screen devices like Samsung (Tizen), LG (webOS), Hisense, Vizio, Chromecast, PlayStation, XBOX, AndroidTV smart TVs and dongles, FireTV, Apple TV (tvOS) and Roku, among others.



Figure 1: Most OTT video services currently support over 10 platforms

The importance of a video player

While delivering premium cross-platform video playback experiences is expected, it remains a really complex challenge. It can be tricky to account for all the video playback variables, including different devices, platforms, operating systems, programming languages, screen sizes, design requirements, and varying network conditions.

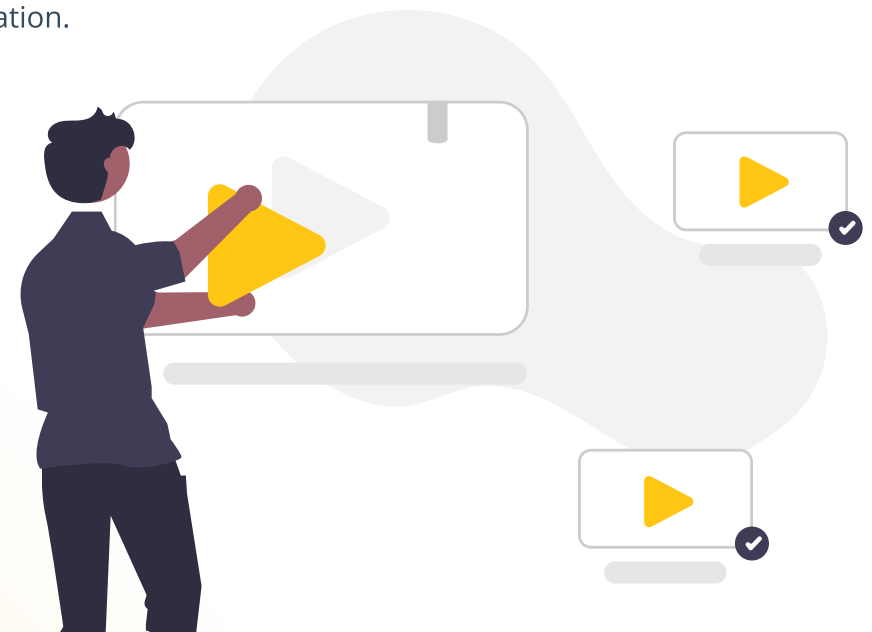
Serving as the “face” of your streaming backend, the video player has a profound impact on user experience. It can effectively mask potential backend issues from viewers, and influence critical parameters for the experience such as channel change times, latency and buffering. Moreover, the video player is heavily integrated with your user interface, or even contains the building blocks for it.

Integration with content protection, analytics, advertising, and other peripheral services further adds to the complexities. The video player plays a central role in security, integrating with authentication and DRM systems. It is also instrumental in driving advertising revenue, integrating with both Client-Side Ad Insertion (CSAI) and Server-Side Ad Insertion (SSAI) systems, ensuring the smooth transition of ads and accurate firing of ad beacons for

tracking, a critical aspect as inaccuracies in ad beacon firing can lead to revenue loss.

Moreover, the video player acts as a vital data source for analytics and measurement systems, offering insights into viewer behavior, content effectiveness, and customer loyalty. Inadequate monitoring can compound these challenges, leaving you blind to critical performance issues, or chasing phantom problems, hindering optimization efforts and impacting viewer satisfaction.

Last but not least, the video player contributes significantly to the overall reliability and resilience of your streaming service, actively participating in content steering and backup stream implementation.



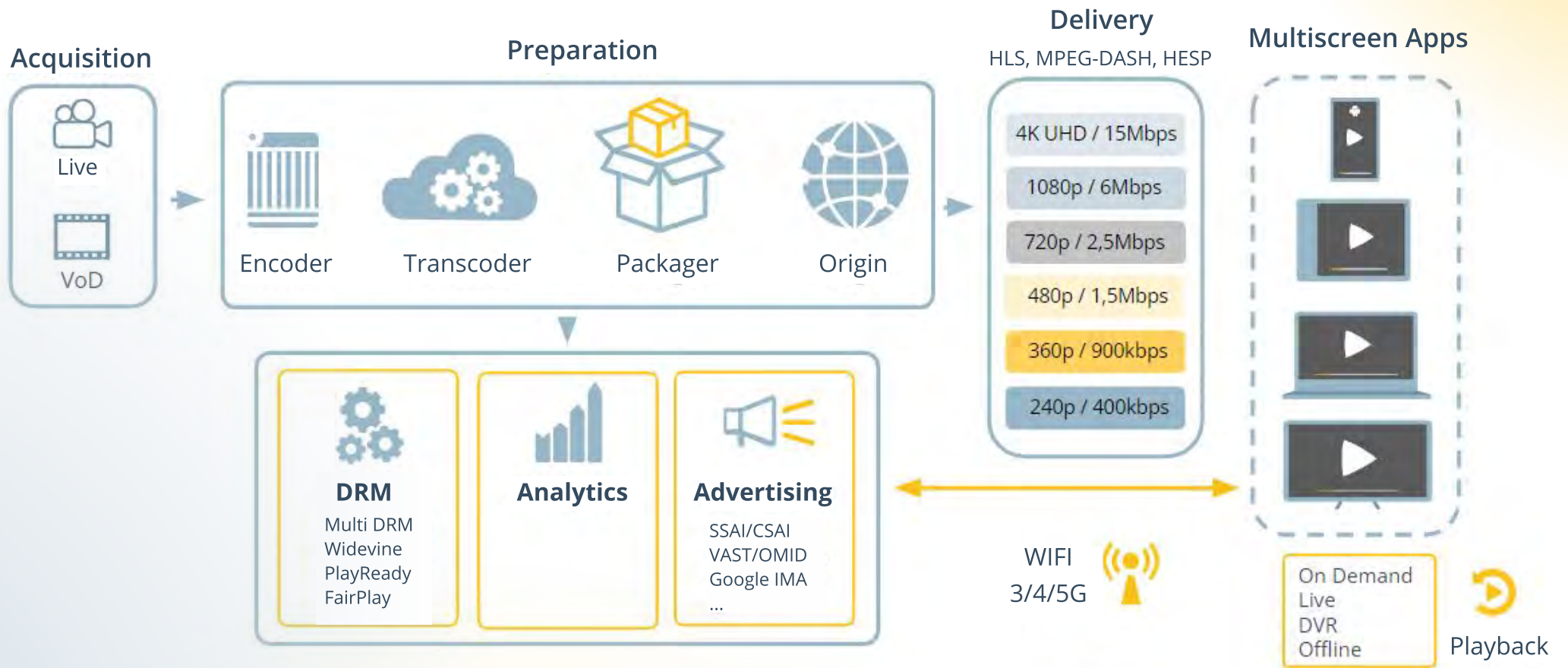


Figure 2: The technical complexity of delivering exceptional video playback experiences cross-platform

Video player evaluation criteria

As the video player plays such an important role, media and entertainment companies either want full control over this crucial component or want to work with a strong knowledgeable partner. When choosing the right video player method for your streaming service involves careful consideration including several key factors:

1. Time-to-market.

Integration is a key consideration, as it determines the time-to-market. It encompasses not only the ease of integrating the player itself but also the incorporation of external services such as ads, analytics, and DRM, and the use of development frameworks such as React Native and Flutter. While some players offer connectors and bridges, thorough testing of these combinations is often overlooked, potentially leading to unforeseen issues, and a prolonged time-to-market.

2. Reliability & testing.

Testing is a critical aspect of ensuring the reliability and functionality of the video player. A robust testing setup is essential, whether conducted internally, externally, by a video player vendor or via open-source communities. A glitchy or

malfunctioning player can lead to a poor viewer quality of experience, a major deterrent for retaining viewers. Viewers expect a video player which just works, and which provides a consistent high-quality video playback behavior cross-platform.

3. Technical support levels.

Bugs and issues are inevitable. When problems arise, it's crucial to know where to seek technical support—whether from your in-house team or an external provider. Swift issue resolution is essential to ensuring a premium experience for your viewers.

4. Maintainability & total cost of ownership.

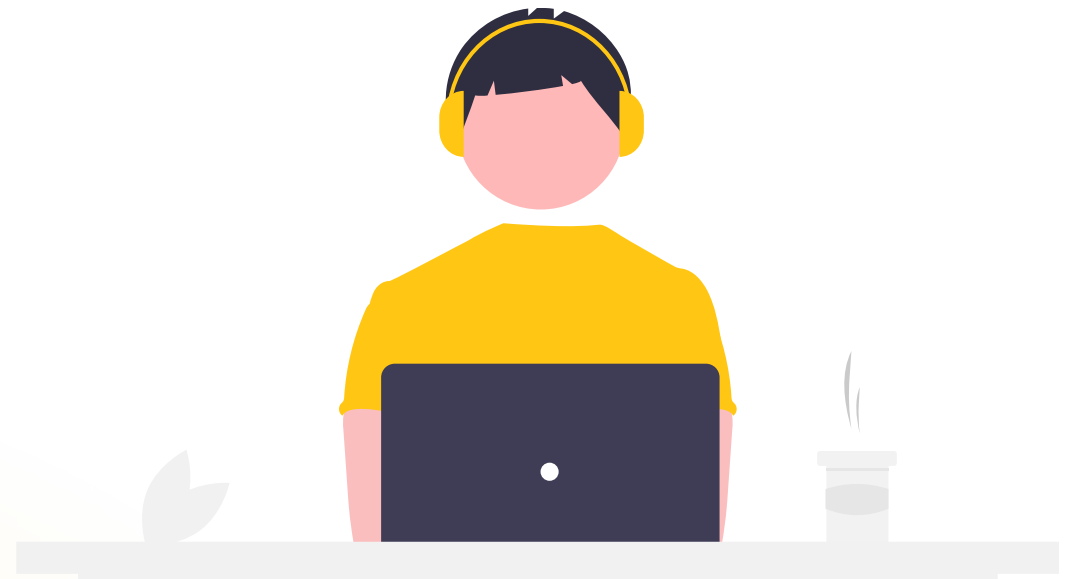
Video player maintenance is often an overlooked but vital aspect. The streaming world is always changing, with new operating systems, platforms, and devices popping up. Putting a video player in place is just the starting point. As streaming technology advances, updates and new features can make things complicated. Remember, everything you create needs ongoing maintenance, which is a big part of the overall cost.








DIY with open source vs. commercial video player approach

Media and entertainment companies usually choose between two video playback approaches:

1. **Building their own video player**
2. **Buying a commercial video player**

Some companies choose the building approach with the **do-it-yourself (DIY) route**, leveraging open source and native video players for cross-platform video playback. Open source players are free, and offer flexibility and customization, while native players, the default for specific platforms, are also cost-free but closed source. There are various open source and native video players, each with its specific platform focus, feature set, and APIs. The below visual shows platform support for common open source and native video players. A more detailed overview of open source and native video players can be found in Annex 2.



	AV Player	dash.js	ExoPlayer	hls.js	Shaka Player	Roku Video Node	video.js	THEOplayer
	-	✓	-	✓	✓	-	✓	✓
	-	⊖	-	⊖	✓	-	⊖	✓
	-	-	✓	-	-	-	-	✓
	✓	-	-	-	-	-	-	✓
androidtv	-	-	✓	-	-	-	-	✓
	-	-	⊖	-	-	-	-	✓
	✓	-	-	-	-	-	-	✓
Roku	-	-	-	-	-	✓	-	✓
 chromecast	-	⊖	-	⊖	✓	-	⊖	✓
VIZIO	-	⊖	-	⊖	⊖	-	⊖	✓
TIZEN SM	-	⊖	-	⊖	✓	-	⊖	✓
Hisense	-	⊖	-	⊖	⊖	-	⊖	✓
HbbTV	-	⊖	-	⊖	⊖	-	⊖	✓
webOS	-	⊖	-	⊖	⊖	-	⊖	✓

- ✓ Officially supported
- ⊖ Compatibility expected

Figure 3: Platform support overview of open source/native video players, and a commercial video player (updated in April 2024)

Alternatively, companies that select the **buying approach** seek a solution that simply 'works', avoiding technical complexities and ensuring peace of mind. Commercial video players typically provide a high viewer quality of experience (QoE) across various devices and platforms through a suite of core playback SDKs designed for cross-platform video playback.

For example, THEOplayer was built from the ground up for high performance and complete control of the AV pipeline, the video playback core of these SDKs is closed source. However, some of the components are open source, including the user interface, connectors for third-party content protection, analytics, and advertising solutions, as well as bridges for development frameworks such as React Native and Flutter.

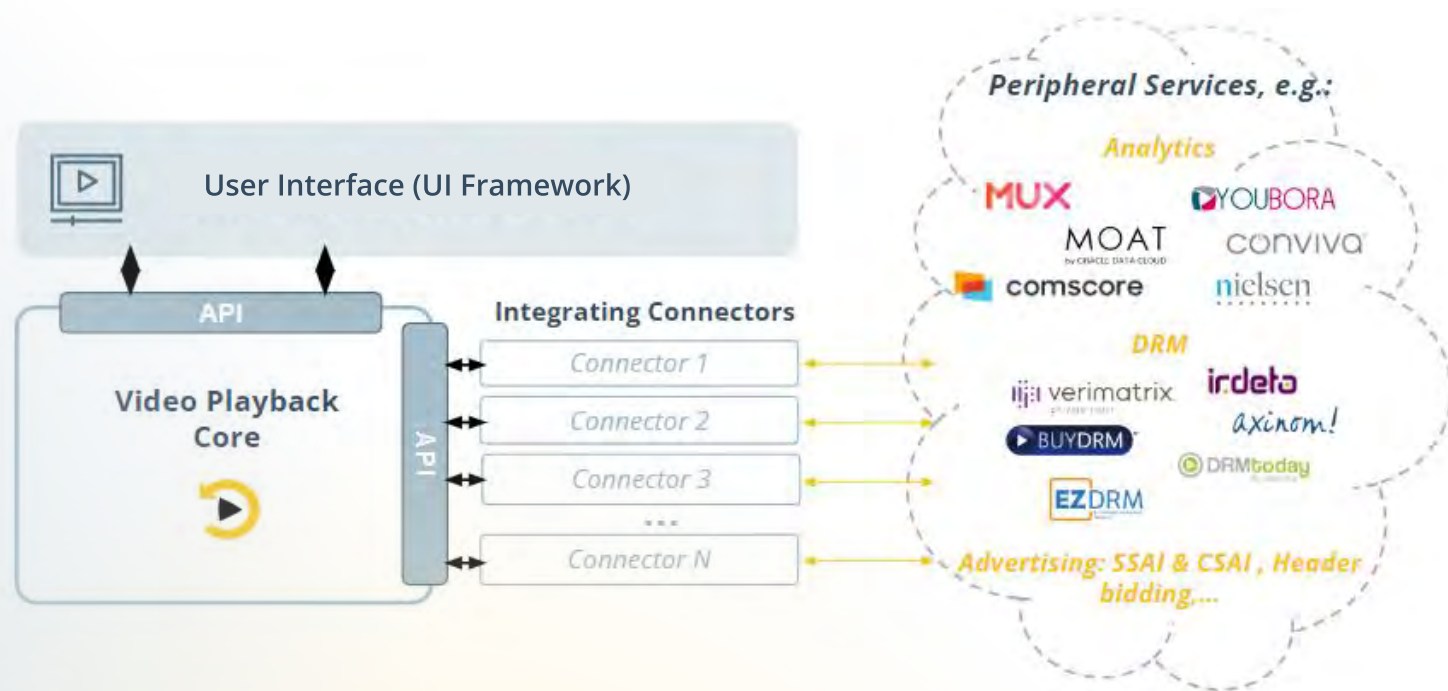


Figure 4: A commercial video player typically consists of a closed source video playback core, and it may additionally contain open source connectors, an open source user interface and open source React Native & Flutter bridges

Pros and cons of both approaches

In this section we'll compare the DIY approach with open source and native video players against the commercial video player approach, weighing their respective pros and cons. The table below provides a summary for each decision criteria, with further elaboration provided in subsequent subsections.




	DIY with open source & native video players	Commercial video players
Time-to-market	 <p>Long time-to-market as each open source & native video player has its own set of APIs and user interface. Custom integrations are needed for DRM, ads and analytics solutions, and also for React Native and Flutter bridges.</p>	 <p>A single set of APIs cross-platform, ready-made DRM, ads and analytics connectors, a comprehensive open source UI, and React Native & Flutter bridges accelerate time-to-market. Note that the availability of these items varies among commercial video players.</p>
Reliability & testing	 <p>Only platforms officially supported are tested by the open source video player teams. The testing focuses on the player only, not the broader ecosystem such as integrations with DRM, analytics and ads solutions. Moreover, open source video players often show black screens or stalls when doing ad transitions.</p>	 <p>Leading commercial video player vendors maintain dedicated test labs to rigorously assess video playback performance for new OS releases across diverse platforms and scenarios. The extent of automated testing and platform coverage may vary among commercial video player vendors.</p>
Technical support levels	 <p>Dependency on community support can cause delays in addressing issues and developing new features. The contributions for open source video players are concentrated on only 1-3 persons.</p>	 <p>Dedicated in-house developer teams specialized in Android, iOS, web/HTML5 and Roku usually offer support, ensuring issues get solved quickly, and new features are proactively implemented.</p>
Maintainability & Total cost of ownership	 <p>Open source and native video players are free, but they bring a high implementation and maintenance cost, resulting in a high total cost of ownership.</p>	 <p>The commercial license fee is typically offset by substantial reductions in implementation and maintenance efforts, lowering the total cost of ownership.</p>

Figure 5: Pros & cons of open source and commercial video players

Time-to-market

Launching an OTT app across platforms can be a lengthy process when relying on multiple open-source and native video players. Each player has its own platform support, feature set, APIs, and developer documentation, complicating the integration process.

Media and entertainment companies opting for a DIY approach with open-source and native video players often face challenges in achieving comprehensive cross-platform coverage. Although open source video players officially support certain platforms, it's essential to recognize that some may only provide community support for select platforms or versions. While these platforms are anticipated to function, they may lack official testing by the video player team.

Furthermore, companies must develop their own connectors to integrate third-party solutions for content protection, analytics, and advertising. This is crucial for business operations, such as ad placement and data insights, or to meet legal requirements for content protection and DRM.

In the context of development frameworks like Flutter and React Native, the complexity is compounded. Here, you need to create

your own bridges for each of the different video players, considering their specific features and connectors for third-party content protection, analytics, and advertising solutions, requiring native development again. For media and entertainment companies relying on a talented pool of Flutter / React Native developers, this means either compromising on functionality or hiring external expertise.

Overall, the various integration requirements, from multiple player APIs to building custom connectors, collectively contribute to extended time-to-market for OTT apps when using open-source and native video players.

In contrast, commercial video players typically offer a unified set of APIs across their SDKs, simplifying integration across various platforms. Commercial solutions often also provide official support for every platform. Unlike open source players that rely on community support or anticipate compatibility for certain platforms, commercial solutions typically undergo comprehensive testing across all platforms. This includes smart TV platforms such as HbbTV, Vizio, WebOS (3.0+), Tizen (2.3+), and FireTV, where thorough investigation, continuous testing, and bug fixes are conducted to guarantee seamless functionality and optimal performance.

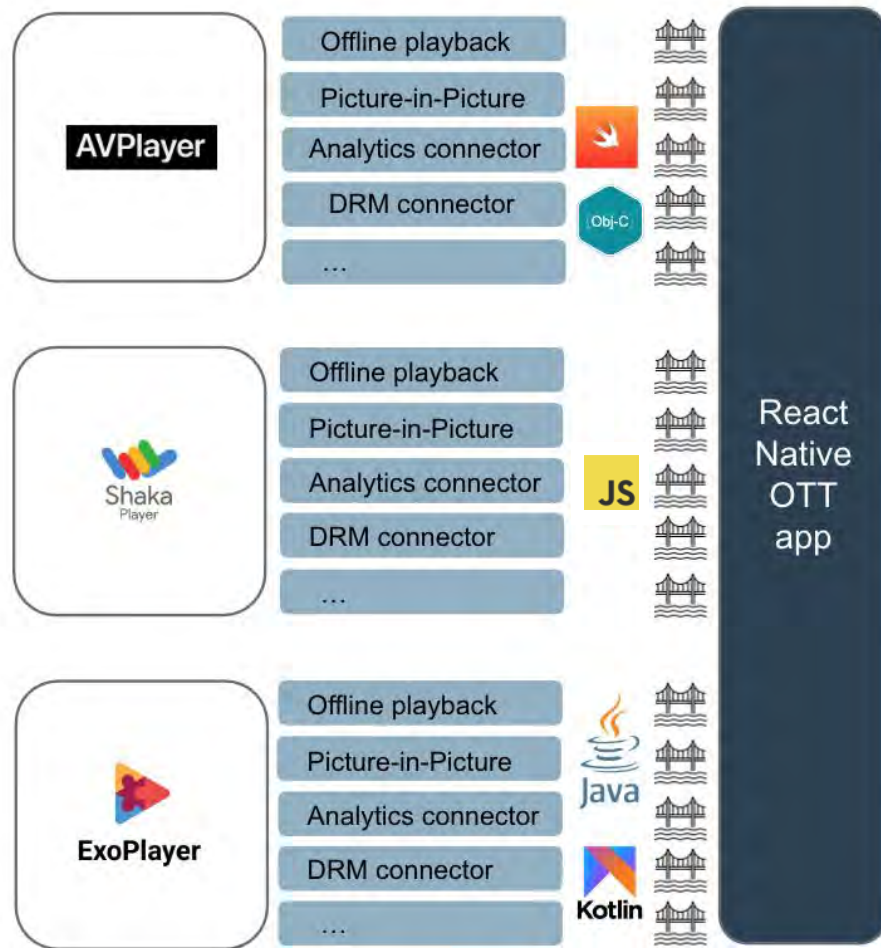


Figure 6: When porting your video pipeline with React Native using open source video players, you often run into native development for premium features and connectors, extending time-to-market.

Moreover, some commercial video players also offer additional benefits that accelerate the time-to-market for your OTT video app.

Firstly, commercial video players can already incorporate open source connectors to ensure integration efficiency and flexibility. These range from connectors for content protection solutions to open source connectors for analytics solutions. Additionally, open source connectors for advertisement solutions are sometimes also available. These connectors play a pivotal role in accelerating the time-to-market of OTT apps, as media and entertainment companies do not have to build these integrations themselves.

Secondly, certain commercial video player solutions also include React Native and Flutter bridges. These bridges enable you to further accelerate time-to-market through easily porting your video pipeline across platforms, and additionally extending compatibility to Web-based platforms such as Samsung's Tizen, LG's webOS, and Vizio. Building React Native and Flutter bridges can be challenging and time consuming, as it requires native development again, so having these bridges available out-of-the-box, and maintained by a commercial video player team is a real time saver.

Thirdly, it's also possible that a commercial video player offers a comprehensive open source UI, typically much more advanced than the UIs provided by open source and native video players.

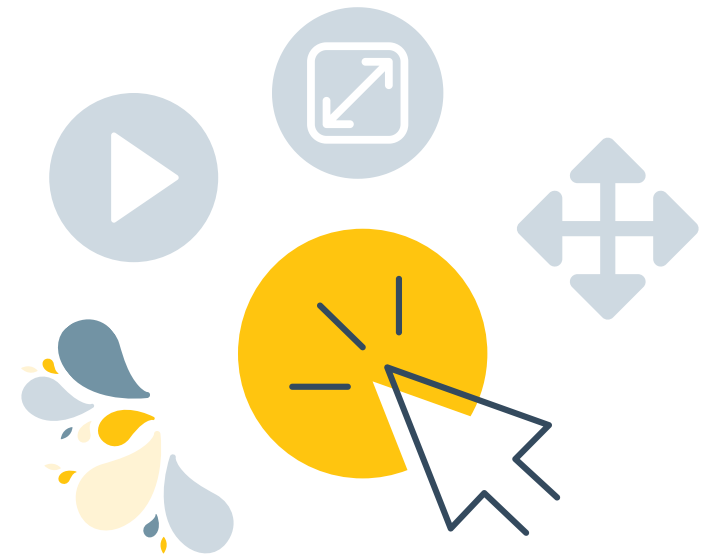
Building a video player user interface from scratch takes time and requires expertise in different platforms, and hence programming languages. This complexity compounds because of different form factors (e.g. small vs big screen), different layouts (e.g. portrait vs landscape mode), different input devices (fingers vs remotes vs mouse & keyboard), different localization requirements (e.g. subtitle languages), as well as accessibility requirements.

Commercial video player vendors which have optimized their open source UI to adapt to these diverse scenarios, really make it easy for media and entertainment companies to build a cross-platform UI.

As an example, THEO's Open Video UI brings an extensive set of available components cross-platform, which makes it easy for media & entertainment companies to customize, saving them from the complexities of creating and maintaining their own video player UI, so that they can focus resources on their own branding.

This open source nature allows media and entertainment companies to easily modify the video player's appearance by customizing each user interface component, including the play button, time display, progress bar, picture-in-picture button, etc.

Despite being open source, the UI is typically actively maintained by the vendor, ensuring ongoing updates and improvements, emphasizing a commitment to keeping the user interface functional across platforms.



Reliability & testing

Open source video players have become popular for their transparency and community-driven development. They offer a solid foundation for media and entertainment companies that desire complete control over their video player's source code to tailor it to their unique requirements. With full access to the code, it's even possible for media and entertainment companies to make changes to the core playback engine, fine-tuning elements like Adaptive Bitrate (ABR) and buffer management for specific devices and markets. However, this path is less traveled, as it demands considerable expertise, in-depth video streaming knowledge, and rigorous testing to guarantee optimal viewer experiences across various platforms.

Testing is a significant part when launching or updating your streaming service. When you're using open source or native video players, the job of testing lands on your shoulders. Supporting multiple platforms adds an extra layer of complexity. How do you effectively test across all these platforms and devices, considering the diverse streaming conditions? It's a time-consuming task that directly impacts viewer experience. Whereas many media and entertainment companies still perform manual testing, either internally or externally, by a third-party vendor, some have already invested in automated testing scenarios.

This is where a commercial video player can excel, as certain commercial video player vendors maintain dedicated test labs where they rigorously assess video playback for new OS releases across various platforms. This extensive testing covers different scenarios, including live streaming, video-on-demand (VOD), restart functionalities, subtitle styling, digital rights management (DRM), license handling, server-side ad insertion (SSAI), and more. The testing also includes combinations with connectors and bridges, to cover for new versions of third-party libraries. A rigorous testing process not only speeds up your time-to-market but also brings you peace of mind and even more confidence.





Figure 7: Commercial vendors can maintain dedicated test labs for rigorous video playback testing

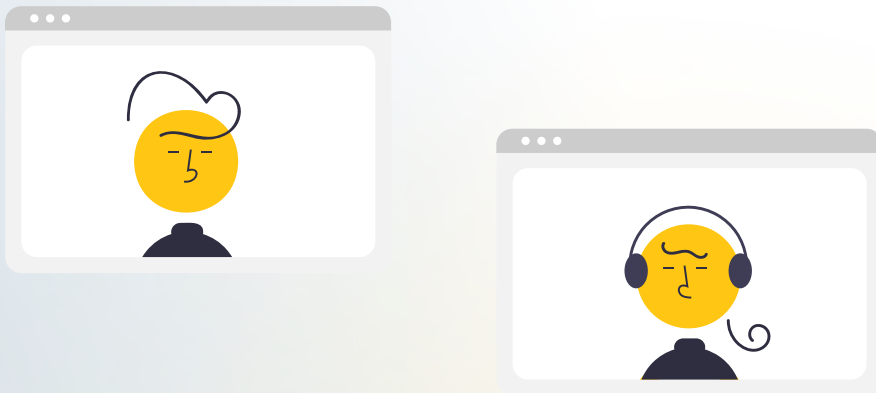
Some commercial video players also excel in optimizing Quality of Experience (QoE) by ensuring fast startup/channel change time, and employing optimized Adaptive Bitrate (ABR) and buffer management algorithms. Most media and entertainment companies opt to leverage these pre-optimized algorithms, recognizing the expertise and thorough testing required to ensure optimal viewer experiences across diverse platforms.

Lastly, the best-of-breed commercial video players also support seamless advertisement transitions. Unlike open-source players, which lack dedicated ownership for seamless integration, challenges may arise when integrating Server-Side Ad Insertion (SSAI) in combination with Digital Rights Management (DRM). These challenges can lead to issues such as black screens or buffering during transitions. However, some commercial video players offer optimized ad insertion capabilities. This ensures smooth transitions, even on older smart TVs and between DRM-protected video content and clear ads. Seamless advertisement transitions are crucial for enhancing advertising CPM, especially with the growing prominence of FAST and AVOD services.

Technical support levels

Relying on the community for bug fixing and handling feature requests is a common challenge associated with open source video players. While these video players offer transparency and community-driven development, the dependency on community support can cause delays in addressing issues and developing new features.

Additionally, for many open source video players, the contributions are very concentrated. Typically, a small group of 1-3 contributors is responsible for over 50% of the commits. This concentration can impact the responsiveness to issues, requiring media and entertainment companies to build internal knowledge about each open source video player, to be able to create their own pull requests when an urgent issue comes up.



Examples:

- For dash.js, the development is managed by DASH-IF on a rotating contract basis. From January to March 2024 about 49 commits were submitted to the project, of which 32 originated from the person supported by that contract.
- Recently ExoPlayer's development on GitHub seems to have slowed, possibly influenced by its integration into the AndroidX library. However, also in this library, the number of commits has decreased.
- While most of the original hls.js developers and supporters over the years have left the project, Apple hired a number of them to maintain the project. The number 1 committer on the project is a bot, which means only 2 people have been actively involved in the last three months.
- 60% of the Shaka Player commits originated from a single person in the period from January to March 2024.
- The majority of the hls.js commits over the same period were performed by 2 contributors.

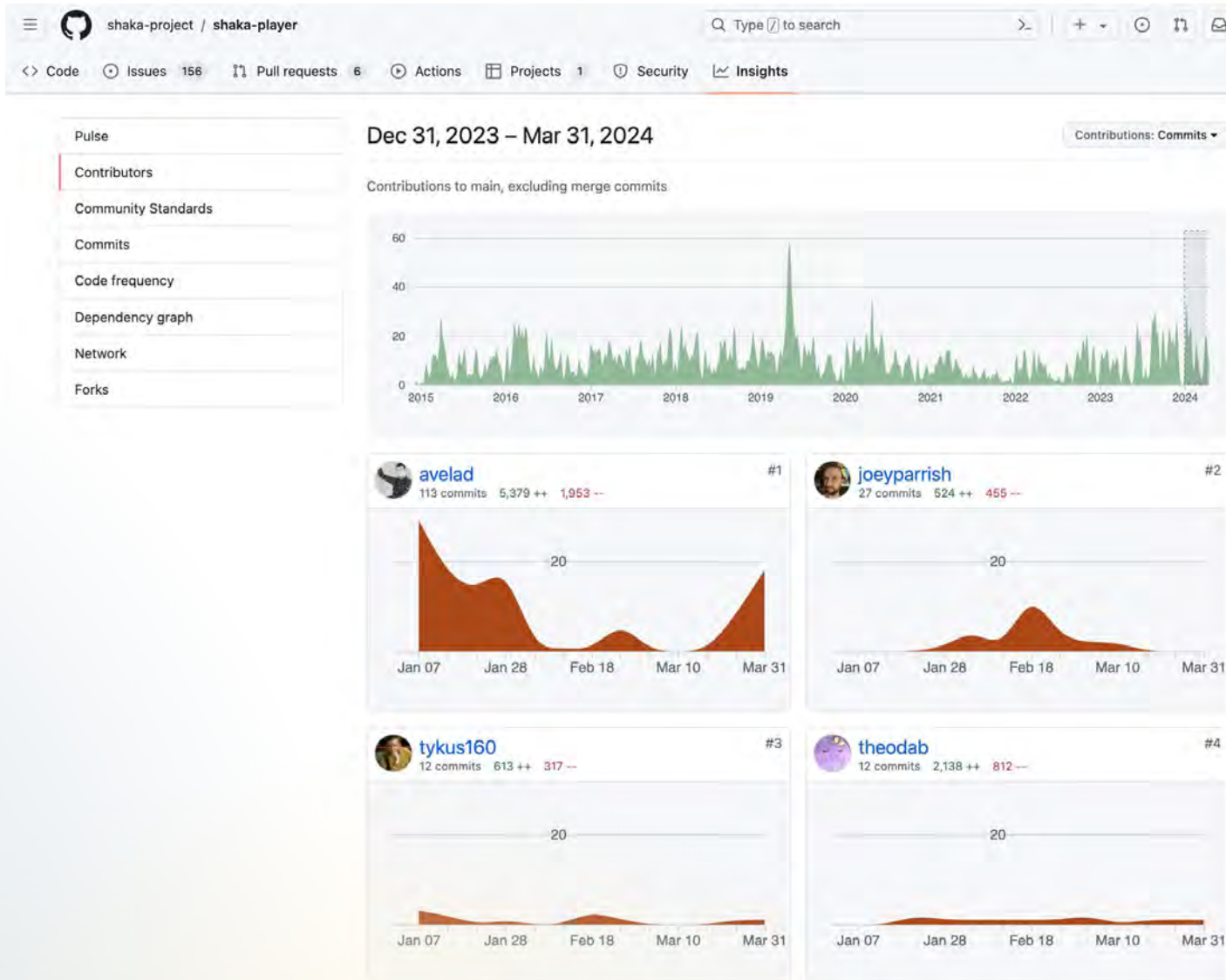


Figure 8: 60% of the Shaka Player commits for the last 3 months (January - March 2024) originated from one person

Conversely, opting for a commercial video player brings the significant benefit of accessing professional support from a team of video experts. With dedicated in-house developer teams specializing in Android, iOS, web/HTML5, and Roku, your bugs and issues get solved faster, effectively minimizing the negative impact.

A commercial video player team stands ready to offer ad hoc assistance and will ensure future evolutions of features and devices. Their experts usually bring end-to-end advisory support, not just for the video player. This brings a dynamic and responsive partnership as your video services evolve.

Moreover, commercial video player vendors often invest in premium features designed to enhance the streaming experience and simplify operations. These include, for example:

- **Extensive features** such as Chromecast and Airplay support, offline playback, background playback, picture-in-picture, Android Media Sessions, and more.
- **Cross-platform framework support** like React-Native and Flutter, enabling a unified codebase for iOS, Android, and Web applications.

- **Alignment with industry trends**, such as EXPO support for React-Native, ensuring compatibility and forward-thinking adaptability.
- **Client-side ad insertion** that seamlessly integrates across all platforms, including smart TVs and Chromecast.
- Integrations with **Server-side ad insertion** platforms to facilitate client-side beaconing and even click-through functionalities.

Maintenance effort & total cost of ownership

Both open source and native video players offer a clear advantage: they are license-free, allowing developers to integrate them into their OTT apps without any upfront costs.

For projects with a singular platform focus, open source video players can be cost-effective. The integration of just one open source or native video player doesn't have the complexities of cross-platform video playback. In contrast, when having to implement multiple open source and native video players, each with its specific platform focus, feature set and APIs, it's important to not only consider the license fee but also the implementation and maintenance complexity and associated effort, to determine the total cost of ownership.

While open source and native video players may seem cost-effective initially, as they come without licence costs, taking a do-it-yourself (DIY) approach for cross-platform video playback typically results in a higher total cost of ownership as the most significant expense in operating a streaming service lies in hiring skilled engineers for development, quality assurance (QA), and design.

Not only the integration effort requires additional FTEs, but the open source video player approach also comes with an ongoing high maintenance burden:

- For instance, if you are supporting over 10 different platforms with different models and versions, you would spend several hours testing each release for every platform. In the end, you may find yourself needing one or more full-time equivalents (FTEs) solely dedicated to testing, definitely for platforms not officially supported by open source video players.
- Maintenance further demands developers who can understand the code for each specific platform. In many cases, this involves having developers with expertise in Android (Java/Kotlin), iOS (Swift, Objective C), web/HTML5 (JavaScript/TypeScript), and Roku (BrightScript) on staff. These resources are needed, at least in part, for solving bugs and issues in time. As a result, this dynamic adds a few more FTEs to the maintenance effort.

- Solving issues is not all, the complexity grows when considering necessary upgrades to new versions of platforms (such as iOS/Android/Tizen releases) and new versions of third-party libraries (Google Cast SDK, IMA SDK, OMID SDK, analytics libraries, etc.). This further increases the need for personnel, adding to the overall costs of the DIY approach.

In contrast, choosing a commercial video player approach includes support from an experienced video player engineering team that develops for the largest OTT streaming services. These experts not only handle the maintenance of the video player core but also oversee open source components like the user interface, connectors, and React Native and Flutter bridges.

Considering that the most substantial streaming service cost lies in hiring experienced engineers for development, quality assurance and design, adopting a commercial video player results in an overall reduction in the total cost of ownership. This allows media and entertainment companies to reassign experienced video engineers from routine video player maintenance tasks to more strategic projects. This not only optimizes the use of experienced video engineers but also empowers them to engage in projects that foster innovation and align with broader organizational goals.

In the media and entertainment industry, there's a growing focus on cost-efficient approaches for video operations. Despite its license fee, a commercial video player brings significant advantages in this respect, by speeding up time-to-market and easing maintenance efforts.

Total Cost of Ownership

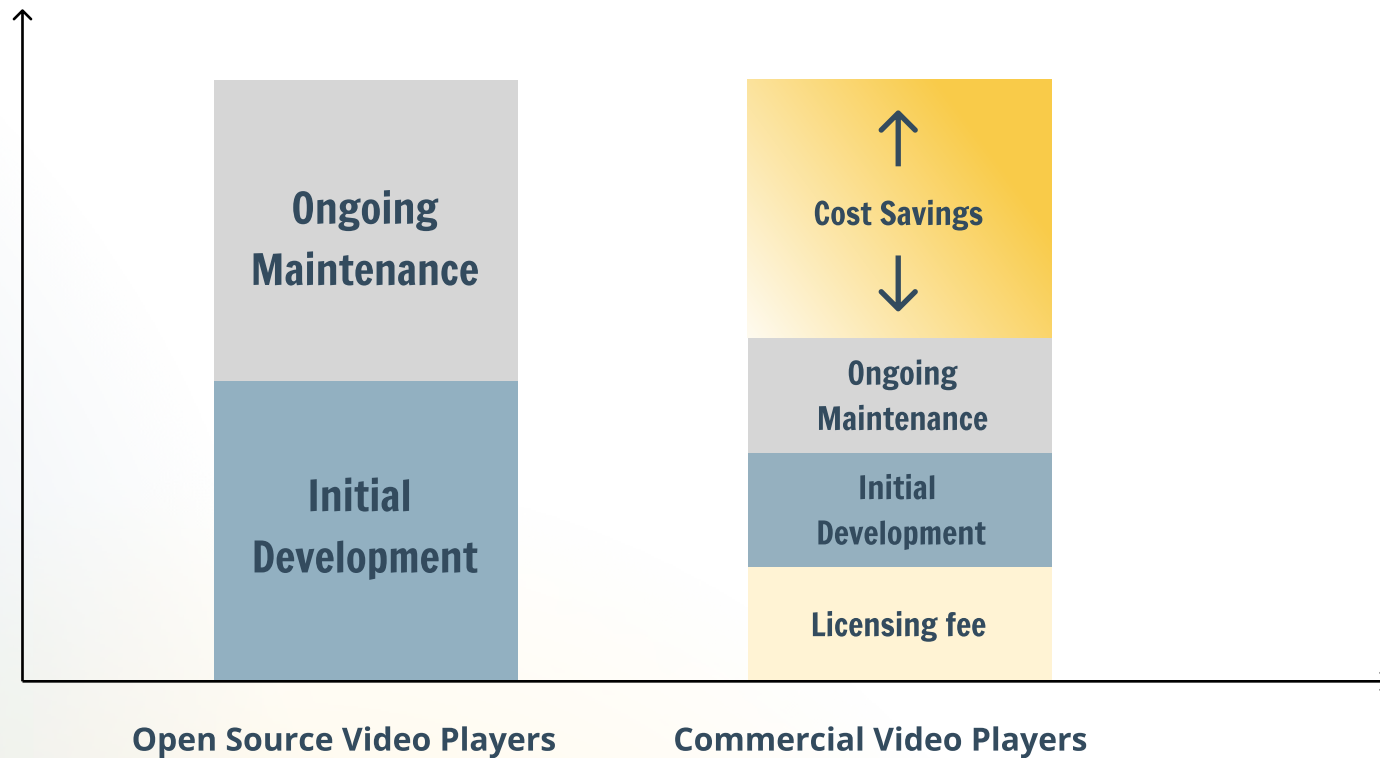


Figure 9: Doing cross-platform video playback with a commercial video player brings a lower Total Cost of Ownership compared to an open source video player approach.

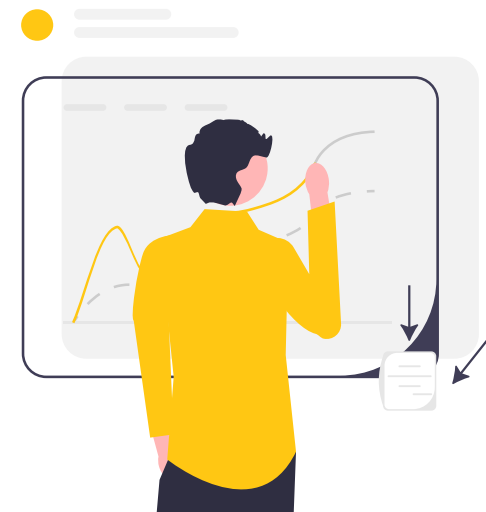
Conclusion

When deciding between a DIY and commercial video player approach, your decision should align with your project's specific requirements and expertise. While the DIY approach may work well for projects with a singular platform focus, the complexities of cross-platform development inevitably lead to higher integration and maintenance effort when going the DIY route. Building your own cross-platform video player based on open source is a product, not a project. You know that you'll be modifying things forever.

A commercial video player brings the **fastest and most cost-efficient** way to deliver premium video experiences cross-platform. Thanks to rigorous cross-platform testing, it brings a high-quality playback solution that simply works. Moreover, it significantly **accelerates time-to-market and minimizes maintenance effort** thanks to pre-made DRM, ads and analytics connectors, premium features, React Native and Flutter bridges, a unified set of APIs across platforms and official support for all platforms. Despite the perceived upfront cost, opting for a commercial video player reduces the total cost of ownership in the long run.

By freeing up in-house video engineers from routine maintenance tasks, a commercial video player also enables media & entertainment companies to focus on strategic initiatives that drive innovation and align with business objectives. This shift not only addresses immediate needs but also supports long-term profitability goals, reflecting the current industry focus on sustainable growth and innovation.

For FAST and AVOD services, a commercial video player will also help **boost advertising revenue** by reaching and monetizing more platforms.



Annex 1: Checklist

Searching for a commercial video playback solution? Below are 20 questions to consider when choosing a video player for your cross-platform OTT app.

- Which platforms does the video player officially support, and does this align with my needs?
- How far back does the version support go for the respective platforms?
- Which technology should be used for these platforms (e.g., Swift, Kotlin, React, React Native, Flutter), and does the player support it?
- Does the video player support all features needed (e.g., offline playback, Android media sessions)?
- Is it possible to ensure code re-use with development frameworks such as React Native to maximize efficiency?
- Is my streaming stack supported (e.g., MPEG-DASH, HLS, CMAF-CTE, LL-HLS, HESP)?
- How can I stand out with my video player user interface, and how easy is it to build that experience cross-platform?
- Does the video player have out-of-the-box support for the user interface, or should I build it from scratch?
- What level of support is provided (e.g., community support, dedicated support team, end-to-end advisory), and does this align with my needs?
- Does the video playback solution simplify and reduce costs for my OTT video service in the future?
- Does the video player vendor regularly maintain the video player, its user interface, bridges, and/or connectors with other solutions?
- How does the video playback solution ensure a high QoE?
- Does the video player vendor perform testing, and which scenarios are covered as part of these tests?
- Does the video player make it easy to measure and gain insights into QoE?
- Which other solutions do I currently have in place, and will it be easy to integrate the video player with them?
- Does the video player have support for DRM, and how easy is it to get started with third-party DRM solutions?
- How can I avoid video player vendor lock-in? If it doesn't work, how fast can I switch or change?
- How will I monetize my service, what is required to maximize that, and how does the video player contribute?
- Could the video player maximize ad revenue through reaching new platforms and/or optimizing ad transitions?
- What makes my OTT video service unique, and how will the video player ensure I can focus on this?

Annex 2: Overview of open source and native video players

AVPlayer

AVPlayer is the default native video player provided by Apple on iOS and tvOS devices. Actively maintained by Apple, AVPlayer is designed specifically for the playback of HLS content. With rich features, it provides support for Low-Latency HLS (LL-HLS) and additional functionalities like offline playback. In addition, it has a UIKit that allows you to easily build a user interface.

Note that AVPlayer exclusively supports Apple's HLS format and its FairPlay DRM. It lacks support for other streaming protocols like MPEG-DASH, and alternative DRM solutions (but those are not available with hardware decryption on iOS anyway).

The logo for AVPlayer, featuring the text "AVPlayer" in white on a black rectangular background.

dash.js

dash.js is the official reference player from the DASH Industry Forum (DASH-IF), designed for playing MPEG-DASH content. It's a JavaScript-based player using Media Source Extensions (MSE) and Encrypted Media Extensions (EME). It officially supports browsers and also works on other Web-based platforms, although MSE/EME-capable smart TVs and gaming consoles have not been tested.

This open source player has features like CMCD, CMAF low latency support, and compatibility with various subtitle formats (TTML, IMSC1, WebVTT). It also has a basic user interface.

The logo for dash.js, featuring a blue grid icon followed by the text "dash.js" in a bold, black, sans-serif font.

ExoPlayer

ExoPlayer, a Google-maintained open source video player for Android, offers platform support for Android mobile devices, Android-based smart TVs, and Android-based set-top boxes. While not officially supported on FireOS, version 5 and above should be compatible.

This feature-rich player includes support for playlists, client-side and server-side ad insertion, DRM-protected playback, and a basic user interface. Additionally, ExoPlayer is equipped to handle various adaptive streaming formats, including MPEG-DASH, HLS, CMAF, and LL-HLS.

The latest version of ExoPlayer is now part of AndroidX Media, residing under a new package name. All future developments will be consolidated within this project.



hls.js

hls.js is an open source JavaScript library that implements an HTTP Live Streaming (HLS) client. It relies on HTML5 video and MediaSource Extensions (MSE) for playback. Specializing in HLS/LL-HLS streams, it does not support MPEG-DASH as a format.

It supports most common browsers and other modern MSE/EME capable devices. However, the compatibility details with Web-based smart TVs and gaming consoles are not specified.

hls.js is equipped with various features, including CEA-608/708 captions, WebVTT subtitles, a basic user interface, as well as alternate audio track rendition.

The hls.js logo is the text "hls.js" in a bold, blue, sans-serif font.

Shaka Player

Shaka Player is an open source video player, formerly developed by Google and currently the default player for Chromecast applications. Operating within an HTML5 environment, it utilizes MediaSource Extensions (MSE) and Encrypted Media Extensions (EME) to play MPEG-DASH and HLS formats.

Compatible with HTML5-based platforms, Shaka Player officially supports common browsers, XBOX One, Chromecast, and Tizen 3.0+. The Shaka Player support matrix also includes platforms like WebOS, Tizen 2.4, and PlayStation 4 & 5, however, these platforms are community supported and untested.

Shaka Player offers features like offline playback, low latency video playback via LL-HLS, and an optional user interface library.



Roku Video Node

The Roku Video Node is the default player for Roku. It uses BrightScript as a programming language, which can present challenges in finding developers familiar with it.

The native video player has a limited API and capabilities. It lacks features such as low latency video playback, and its support for streams is highly specific, leading to restrictions on HLS/DASH versions, GOPs, and segment durations. Additionally, it has its own approach for generating thumbnails, and steering Adaptive Bitrate (ABR) algorithms or buffers is not possible with this player.

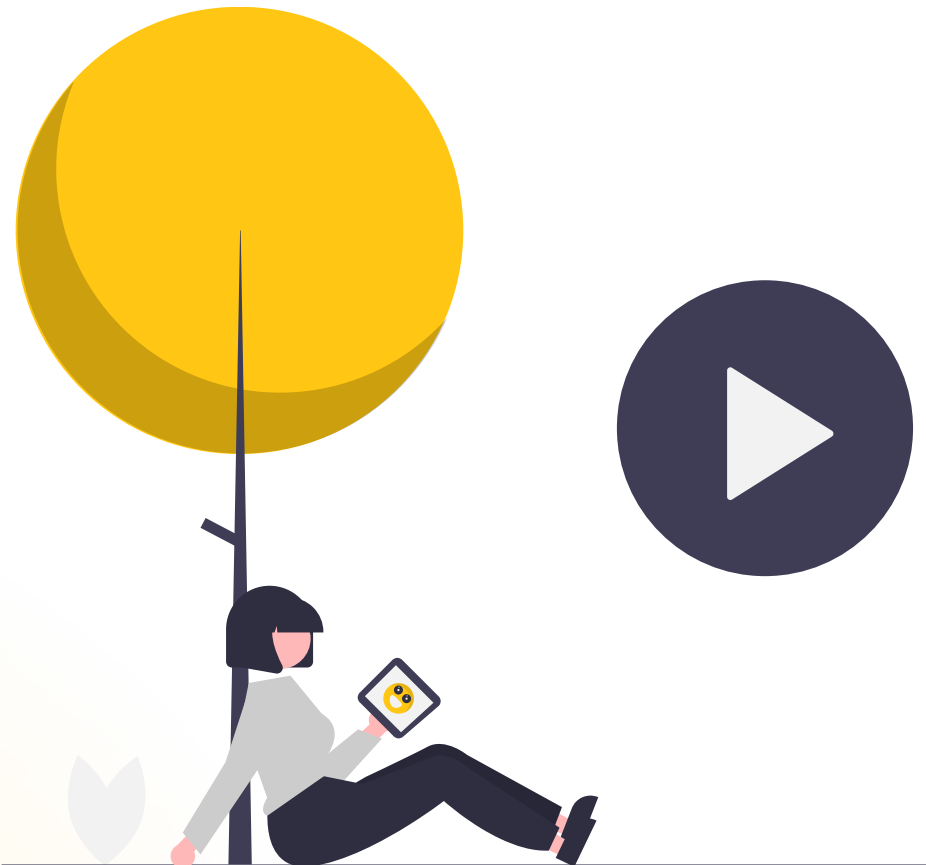


video.js

Video.js is an open source HTML5 video player that supports video playback on desktop and mobile browsers. Additionally, it also supports web-based smart TVs and other platforms, although specific details about supported devices are not provided. Launched in mid-2010, the project was mainly sponsored by Brightcove.

What sets video.js apart is its modular architecture, with many functionalities separated into plugins. This modular design allows developers to share customizations, incorporating features like thumbnails, VR support, and Chromecast integration.

In terms of format support, video.js supports both (LL-)HLS and DASH playback.





DIY with Open Source vs. Commercial Video Player Approach

WHITEPAPER

About THEO Technologies

At THEO Technologies, we are shaping the future of entertainment by providing high-quality video streaming technology. Our mission is to simplify video operations, empowering developers to easily integrate high-quality video into their applications.

WWW.THEOPLAYER.COM

INTERESTED IN LEARNING MORE ABOUT BUILDING YOUR VIDEO PLAYER STRATEGY?

Get in contact with one of our video experts
WWW.THEOPLAYER.COM/CONTACT-US

