Kaltura player architecture

Content updated: 09/01/2022

This article is designated for all users.

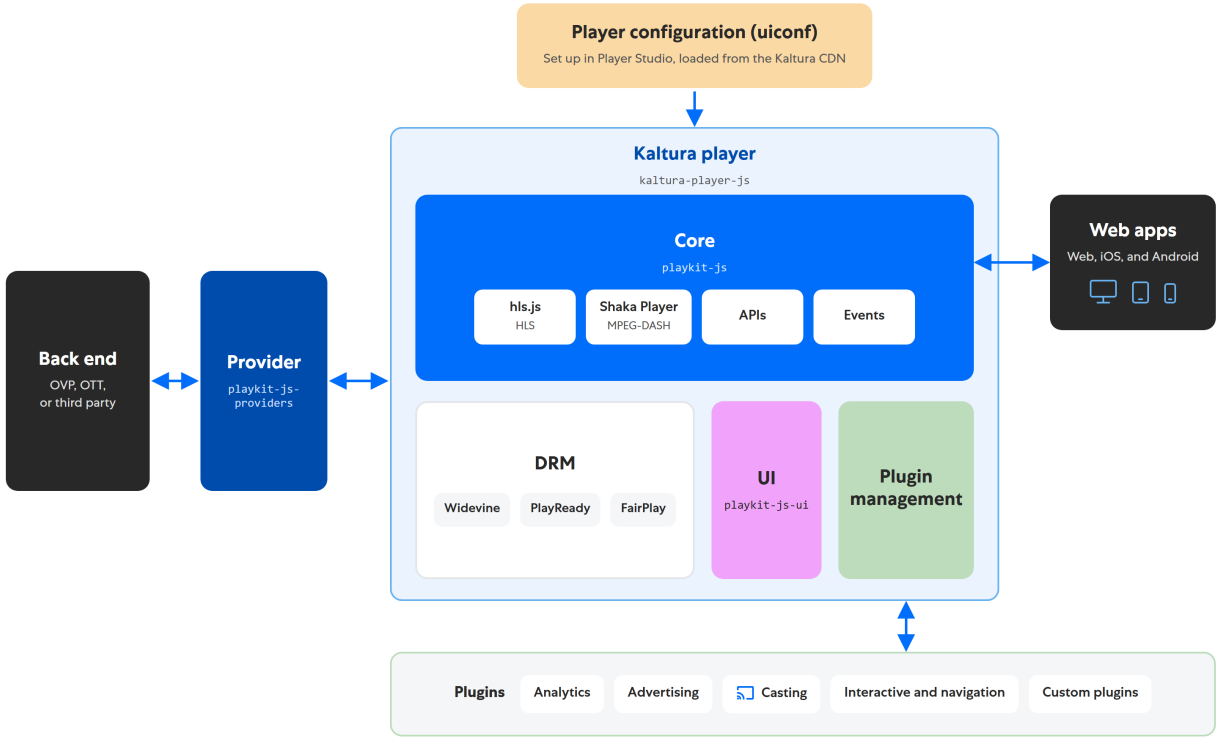
About

The Kaltura player is built from several components, and each one handles a specific part of playing media, such as getting the media from the back end, playing it, or showing the controls.

Knowing how these components work together helps you customize the player, build plugins, or troubleshoot playback.

Architecture overview

The diagram shows how the player's components work together.



The sections below describe each component, what it does, and how it works with the other components.

Components and packages

Copyright © 2025 Kaltura Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this document constitutes acceptance of the Kaltura Terms of Use and Privacy Policy.

Each component in the diagram is a separate open-source JavaScript package.

Component	Package
Player wrapper, which brings the other components together	`kaltura-player-js`
Core (playback engine)	`playkit-js`
Provider (back-end integration)	`playkit-js-providers`
UI	`playkit-js-ui` , built with Preact

Player configuration (uiconf)

When you load the player from the Kaltura CDN, the embed URL includes the ID of a player configuration, called a 'uiconf'. You create and edit this configuration in Player Studio. The embed URL uses this format.

```
https://cdnapisec.kaltura.com/p/{partnerId}/embedPlaykitJs/uiconf_id/{uiconfId}
```

The player loads the uiconf JSON configuration before it starts. The configuration sets which plugins, UI elements, and playback rules are turned on.

Provider

The **provider** is the part of the player that gets a video's information from Kaltura's servers, such as where its files are and whether the viewer can watch it, before playback starts. It communicates with the back end through API requests, and turns the data it gets into media objects that the player can play.

The **back end (BE)** is the server where your media is stored. The Kaltura player can work with a Kaltura back end (OTT or OVP) or with a third-party back end.

How the provider gets media

1. The web app tells the provider which media asset to play, using the asset's media ID.
2. The provider sends a request to the back end for data about that media asset. If authentication is needed, the request includes a `KalturaSession` (KS) token.
3. The back end returns the data to the provider.
4. The provider builds two objects: a media object that the core can play, which can contain media or playlist information, and an error object, which can contain an error code. Only one of them is populated at a time. If there's no error, the media object is

Copyright © 2025 Kaltura Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this document constitutes acceptance of the Kaltura Terms of Use and Privacy Policy.

populated and the error object is null. If there's an error, the media object is null and the error object is populated.

5. The provider passes both objects to the core. If the media object is populated, the core plays the media. If the error object is populated, the core registers an error event, which the web app can handle and respond to.

Core

The core handles media playback. It creates an HTML5 video element, which the browser shows on screen. The core uses two libraries to play different streaming formats:

- hls.js for HLS
- Shaka Player (by Google) for MPEG-DASH

The core also exposes APIs that web apps and plugins use to work with the player, including APIs for playback, the UI, player configuration, and advertising.

It also exposes the player's events. Web apps, plugins, and custom UI frameworks can listen for these events and run actions in response.

How the core plays media

1. When the core gets a media or playlist object from the provider, it uses source selection logic to choose a playback library. The choice depends on the available media formats, configurable priorities, and the browser.
2. The core creates a window in the web app that displays the media or playlist.

Throughout the player's lifecycle, the core registers events, for example when the media loads, when it starts playing, or when an ad break begins. Event listeners can detect these events and trigger a response in the web app or a plugin.

DRM

The Kaltura player supports digital rights management (DRM) with Widevine, PlayReady, and FairPlay. How DRM content plays depends on the browser:

- **Desktop browsers, such as Chrome, Firefox, and Edge:** Widevine and PlayReady content plays through Encrypted Media Extensions (EME). Shaka Player manages the process and communicates with the browser's Content Decryption Module (CDM).
- **Safari on iOS and macOS:** Apple doesn't allow third-party decryption through Media Source Extensions (MSE) and EME in the same way. So the Kaltura player hands FairPlay playback to Safari's native media pipeline (AVFoundation and

Copyright © 2025 Kaltura Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this document constitutes acceptance of the Kaltura Terms of Use and Privacy Policy.

WebKit), and feeds it the DRM license and certificate.

How protected content plays

If the media is encrypted, the Kaltura player detects this during source selection, described in the Core section. On desktop browsers, it passes the media to Shaka Player, which gets a DRM license from a DRM server. Shaka Player doesn't decrypt the content itself. The browser's CDM decrypts it, using keys protected by the device hardware or operating system.

For more information, see Digital Rights Management (DRM).

UI

The user interface (UI) is a customizable layer of interactive buttons displayed over the video. The buttons call the player's APIs to control playback functions, such as play, pause, seek, and closed captions.

You can extend the UI framework with custom UI elements. You can also turn off the player's UI entirely and build your own, using event listeners and the player's APIs.

Plugin management

The Kaltura player supports several pre-built plugins, and it also supports custom plugins.

- **Analytics:** KAVA, Google Analytics, Youbora, and Comscore
- **Advertising:** IMA and IMA DAI
- **Casting:** Chromecast
- **Interactive and navigation features:** Interactive Video Quizzes (IVQ), Transcript, Chapters (navigation), Dual-Stream,Live Q&A, and chat for virtual events

Each plugin monitors the player's lifecycle and responds to player events.

How plugins respond to events

Plugins can listen for player events from the core, such as ad events, and respond to them, for example by sending data to an analytics system. This happens in parallel with the player's other functions, such as video playback.

Copyright © 2025 Kaltura Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this document constitutes acceptance of the Kaltura Terms of Use and Privacy Policy.