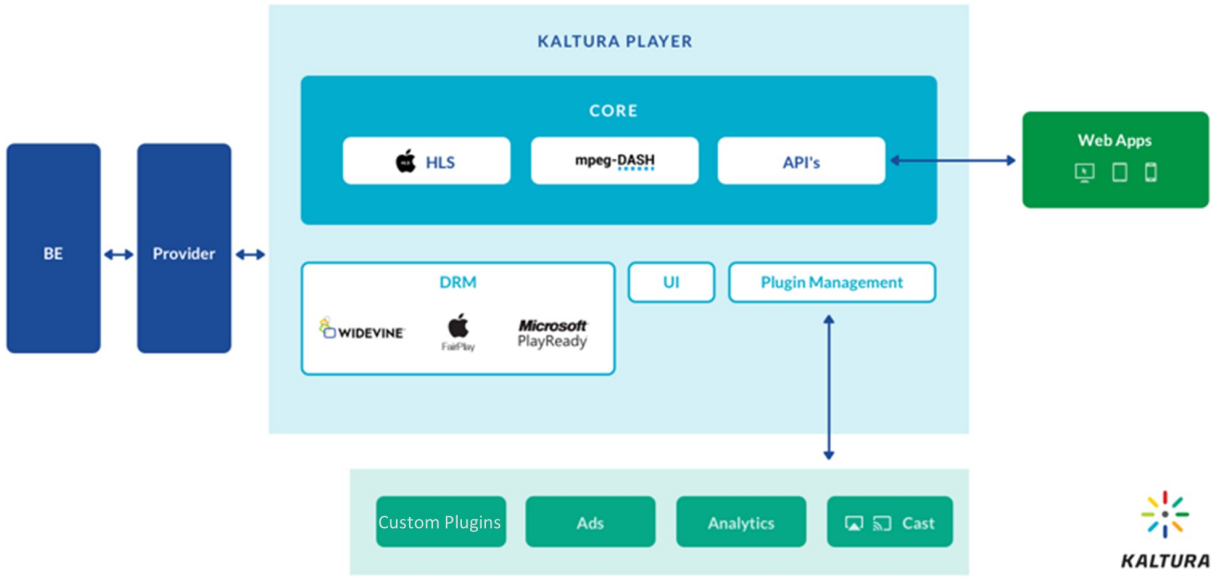
Kaltura player architecture

Last Modified on 05/23/2024 5:25 pm IDT

Overview

Kaltura Player is composed of several modules, each providing a specific function. In this article, we will describe each of these modules, what they do, and how they interact with other Kaltura Player modules.

Player architecture diagram



Back-end (BE) and provider

The **Back-End** (BE) is where the media content is stored. The Kaltura Player may be used with one of Kaltura's back-ends (OTT back-end or OVP back-end), or with a third-party back-end.

The **Provider** is a Kaltura Player module that allows the Kaltura Player to communicate with Kaltura's back-ends via API requests and assembles the obtained data into playable media objects.

How does it work?

The Provider receives input from the Web App specifying which media asset is to be played (specified via its Media ID). The Provider then sends a request to the Back-End,

requesting data about the specified media asset. If required, it also provides authentication (i.e. a *KalturaSession* token) within the request.

The Back-End returns the data about the requested media asset to the Provider. Using the acquired data, the Provider builds two objects: a media object that the Core can play (may contain media or playlist information), and an error object (may contain an error code). Only one of these objects is populated at any given time – if there is no error, then the media object is populated and the error object is null; if there is an error, then the media object is null and the error object is populated.

Lastly, the Provider feeds these two objects to the Core for either (1) playback, or (2) registering an error event (which can then be handled and reacted to by the Web App). Which of these operations occurs depends on which of the two objects is populated.

Core

The **Core** of the Kaltura Player handles playback of media content. It creates an HTML5 video element, which the browser displays as a video on-screen. This Core integrates two libraries for playback of different video streaming formats: (1) HLS.JS for HLS playback, (2) Shaka Player for MPEG-DASH playback.

The Core also exposes APIs for interfacing with the Kaltura Player, including APIs related to playback, UI, Player configuration, advertisements, and many more. This allows front-end Web Apps and plugins to interface with the Player.

Lastly, it exposes the Kaltura Player events that can be detected using an Event Listener. This allows you to detect specific events within the Kaltura Player and trigger actions in response, and can be used by Web Apps, plugins, custom UI frameworks, and more.

How does it work?

When the Core receives a media or playlist object from the Provider (see above), the Core implements source selection logic to determine which library to use depending on the available media formats, configurable priorities, and the web browser used.

The Core then generates a window in the Web App that displays the content (either media content or a playlist).

At various stages of the Kaltura Player's life-cycle, it registers events (e.g. when the media has loaded, when it's playing, when an ad break has begun, etc.). Event Listeners can detect these events and trigger a response by the Web App or plugin.

DRM

DRM is supported by the integrated Shaka Player library. DRM is supported for

Widevine, PlayReady, and FairPlay; however, DRM is NOT supported for a browser's native player.

> ⚠ **NOTE**: On iOS devices, the Kaltura Player uses Safari's native player and feeds it the DRM License and Certificate.

How does it work?

If the media to be played is encrypted, the Kaltura Player recognizes this during source selection (see "Core", above) and passes it to Shaka Player. Shaka Player then obtains a DRM license from a DRM Server, extracts a content encryption key from the DRM license, and decrypts the media content.

UI

The **UI** is a customizable layer of interactive buttons that are displayed over the video. These buttons trigger API calls to the Kaltura Player, which control various playback functions (e.g. Pause, Play, Seek, Closed Captioning, etc.).

It is possible to extend the UI framework to include custom UI elements. It's even possible to disable the Kaltura Player's UI entirely and create your own UI, making use of Event Listeners and the Player's APIs to interface with the Kaltura Player.

Plugin management

The Kaltura Player supports several pre-built **Plugins** that provide Analytics (*KAVA, Google Analytics, Youbora* & *Comscore*), Advertising (*IMA* & *IMA DAI*), and casting (*Chromecast*) functionalities. It also supports custom plugins.

Each plugin is responsible for monitoring the lifecycle of the Kaltura Player and reacting appropriately to Player events.

How does it work?

Plugins can register Kaltura Player events from the Core (e.g. ad events) and react accordingly (e.g. send data to an analytics system). This occurs in parallel to the Kaltura Player's other functions, such as video playback.

[template("cat-subscribe")]