Prebid Mobile Overview

Prebid Mobile is an open-source library that provides an end-to-end header bidding solution for mobile app publishers. Prebid Mobile libraries are available for iOS and Android.

Benefits and Features

The Prebid SDK offers the following benefits:

- **Transparent, open-source header bidding solution**. A single integration point with Prebid Server. This rendering option enables direct access to more mobile ad buyers.
- **Monetization without an ad server**: Publishers who do not have a direct sales team or have no need for an ad server can still access Prebid's mobile demand stack. Publishers will be able to render ads directly without relying on any third-party SDKs.
- **Reduced ad delivery latency**: The rendering module enables Prebid SDK to render ads immediately when demand is returned from Prebid Server or when receiving the render signal from an ad server. The rendering process vastly reduces ad delivery speeds.
- **Less infrastructure**: The rendering API does not rely on Prebid Server's cache, reducing the cost and utility of Prebid Server's cache functionality.
- **Less discrepancy**: Having control of the rendering process provides the potential to reduce discrepancy by having ads instantly available (fewer http calls, less infrastructure, less setup). This control enables the publisher to follow open and transparent industry standards or custom requirements from buyers.
- **Framework support**: Full support of:
  - SKAdNetworks and similar frameworks.
  - MRAID 3.0 support.
- **Flexible Ad Measurement**: Controlling the rendering and Open Measurement process allows publishers to potentially configure any measurement provider in a transparent and open-source process. Prebid SDK will eventually be IAB Open Measurement certified.
- **Ad Server Support**: Prebid SDK can be integrated with any potential monetization stack, such as Google Ad Manager.
- **Multiformat Ad Unit**: The rendering engine will enable Prebid SDK to display any bid format in the given inventory regardless of primary ad server capabilities.
- **Support of Custom Ad Servers**: With the rendering engine, Prebid SDK can work with any ad server. It currently supports Google Ad Manager, AdMob, and MAX.

Integration

Prebid SDK supports the following integration scenarios:

- **Custom Or No Mediation**: When there is no primary ad server the SDK renders the winning bid once it is available.
- **Using a Primary Ad Server**: There are three scenarios available for integration with a primary ad server.
  - **Original API**: In this case, the SDK plays a transport role in the bid from Prebid Server to the primary ad server. The primary ad server conducts an internal auction and the winning bid is rendered with Prebid Universal Creative (PUC) functions.
  - **Rendering API:** Prebid SDK detects when a Prebid line item wins the ad server auction and renders the cached bid in the app's web or video view.
  - **Mediation API:** A Prebid adapter detects when a Prebid line item corresponds to the winning bid and renders the cached bid in the app's web or video view.

  In all scenarios, Prebid SDK leverages Prebid Server for ad demand.

  The following chart shows which Prebid SDK API is used for each ad server:

Ad Server	Original API	Rendering API	Mediation API

No Ad Server		✓	
Google Ad Manager	✓	✓	
Ad Mob			✓
MAX			✓

Platform Integration

Android

Follow these steps to integrate the Prebid SDK on Android apps:

1. If integrating with an ad server create line items specific to that server’s rendering process:

Line items for the rendering API are unique and are not related to the standard Prebid SDK line items

- GAM Original API
- GAM Rendering API
- AdMob
- MAX

2. Integrate Prebid SDK into your project.
3. Add prebid’s ad units to your app’s monetization scenario:

- GAM Original API
- Custom in-app bidding integration without primary ad server.
- GAM Rendering API as a primary ad server.
- AdMob as a primary ad server.
- AppLovin MAX as a primary ad server.

4. Actualize the integration and targeting properties.

iOS

Follow these steps to integrate the rendering API into iOS apps:

1. If integrating with an ad server create line items specific to that server’s rendering process:

Line items for the rendering API are unique and are not related to the standard Prebid SDK line items

- GAM Original API
- GAM Rendering API
- AdMob
- MAX

2. Integrate Prebid SDK into your project.
3. Add prebid’s ad units to your app’s monetization scenario:

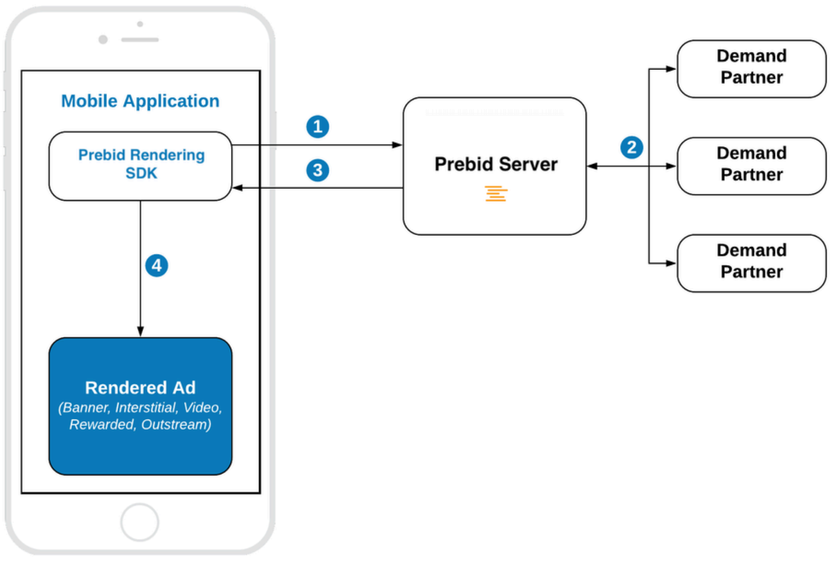
- GAM Original API
- Custom in-app bidding integration without primary ad server.
- GAM Rendering API as a primary ad server.
- AdMob as a primary ad server.
- AppLovin MAX as a primary ad server.

4. Actualize the integration and targeting properties.

Ad Request and Response Flow

The following sections describe each ad request and reponse flow for the different ad server options.

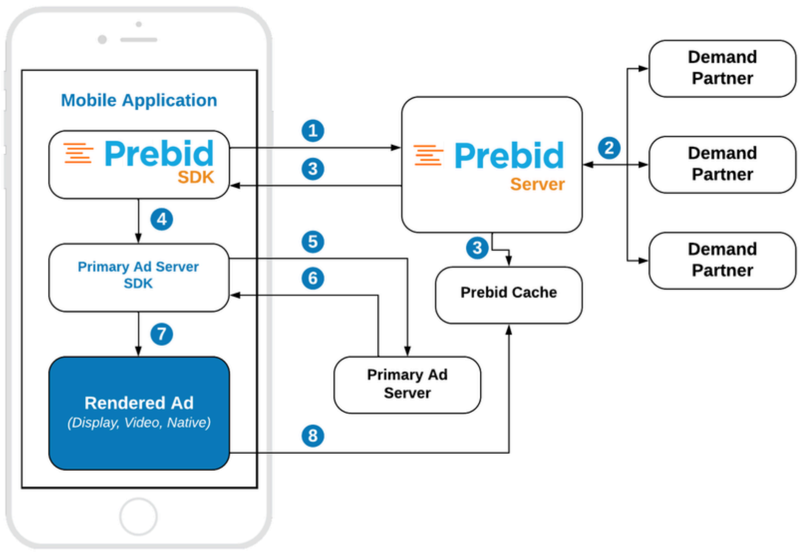
With No Ad Server



1. The Prebid SDK sends the bid request to the Prebid Server.
2. Prebid Server runs the header bidding auction among preconfigured demand partners.
3. Prebid Server responds with the winning bid.
4. The rendering module renders the winning bid.
5.

With An Ad Server's API

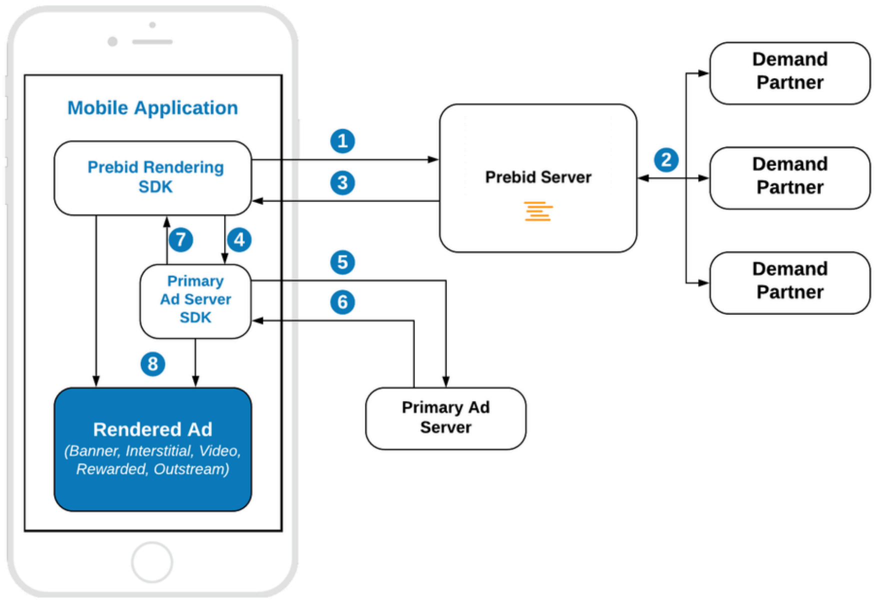
Supported Ad Servers: GAM



1. Prebid Mobile sends a request to Prebid Server. This request consists of the Prebid Server account ID and config ID for each tag included in the request.
2. Prebid Server constructs an OpenRTB bid request and passes it to the demand partners. Each demand partner returns a bid response to Prebid Server. The bid response includes the bid price and the creative content.
3. Prebid Server sends the bid responses to Prebid Mobile.
4. Prebid Mobile sets key/value targeting for each ad slot through the primary ad server mobile SDK.
5. The primary ad server SDK sends the ad request, enriched with targeting keywords, to the primary ad server.
6. The primary ad server responds with an ad. If the line item associated with the Prebid Mobile bid wins, the prebid passes its PUC functions to the ad server's SDK.
7. The primary ad server SDK starts compiling the received ad markup.
8. The PUC fetches the creative content of the winning bid from the Prebid cache and renders it.

With an Ad Server Using Prebid's Rendering API

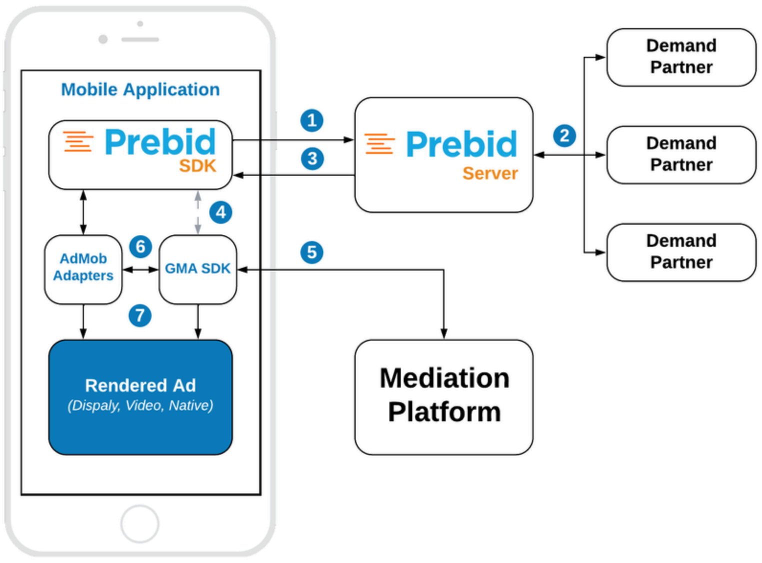
Supported Ad Servers: GAM.



1. The rendering module sends the bid request to the Prebid server.
2. Prebid server runs the header bidding auction among pre-configured demand partners.
3. Prebid Server responds with the winning bid that contains targeting keywords.
4. The rendering module passes the targeting keywords of the winning bid to the primary ad server SDK.
5. The primary ad server SDK sends the ad request to the primary ad server.
6. The primary ad server responds with an ad or mediation chain.
7. The winning ad meta information is passed to the rendering module.
8. Depending on the ad response, the rendering module renders the winning bid or allows the primary ad server SDK to show its own winning ad.

With Ad Server Using Prebid's Mediation API

Supported Ad Servers: AdMob, MAX.



1. The Prebid SDK sends the bid request to the Prebid server.
2. Prebid server runs the header bidding auction among pre-configured demand partners.
3. Prebid Server responds with the winning bid that contains targeting keywords.
4. [OPTIONAL] The rendering module passes the targeting keywords of the winning bid to the primary ad server SDK.
5. The primary ad server SDK sends the ad request to the primary ad server and responds with a mediation chain.
6. If the mediation item contains the name Prebid Adapter it instantiates the respective ad format class.
7. [OPTIONAL] The Prebid Adapter checks if the line item’s targeting keywords match the bid targeting keywords.
8. The Prebid Adapter renders a winning bid which is cached in the SDK.

Note: Passing the targeting keywords to the ad server depends on the server’s ability to target line items. If the server doesn’t provide such a feature, Prebid SDK doesn’t enrich an ad request with targeting info. Activation of a line item with the proper price still works.

Additional references

- Deep Links Support
- Impression Tracking