Prebid Mobile 3.0.0

Contributors: Jono Sligh, Alexander Savelyev, Mike Mullin, Yuriy Velichko, Chris Huie, Christian Janelli

Last Updated: Apr 5, 2024

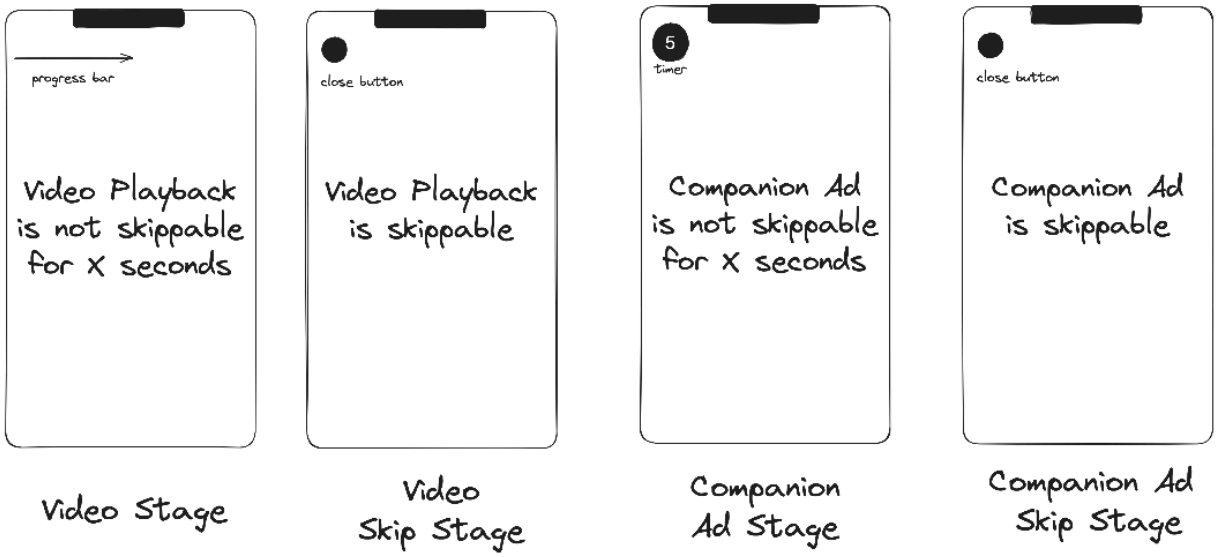
Table of Content:

Overview

The goal of Prebid Mobile 3.0.0 is to attract mobile-first demand. The Prebid Mobile team and PMC will be implementing ad-related features for the modern Video UX such as VAST 4.0 support. Relevant iOS operating system ad-related features such as StoreKit will be included in 3.0.0. This new version will also allow Prebid SDK to operate stand-alone in respect to rendering as well as with 3rd party SDKs for ad rendering. This version will also include a cleanup of accumulated deprecated interfaces and methods. There is also an effort underway to fully overhaul and update the Prebid Mobile SDK documentation.

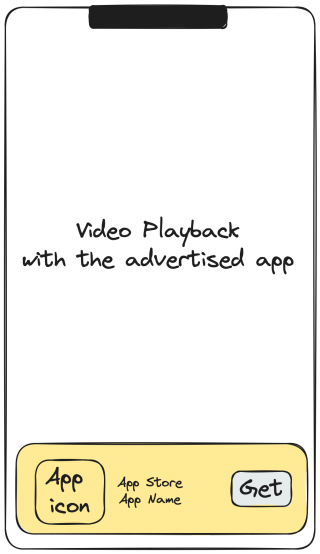
Video UX

Modern video UX on iOS and Android will include the following: static & playable (MRAID) endcards, progress bars, video playback and endcard unskippable for x seconds, triggering relevant events, signal in the bid request & interpret bid response, and VAST support up to 4.0.
[IAB VAST Documentation]



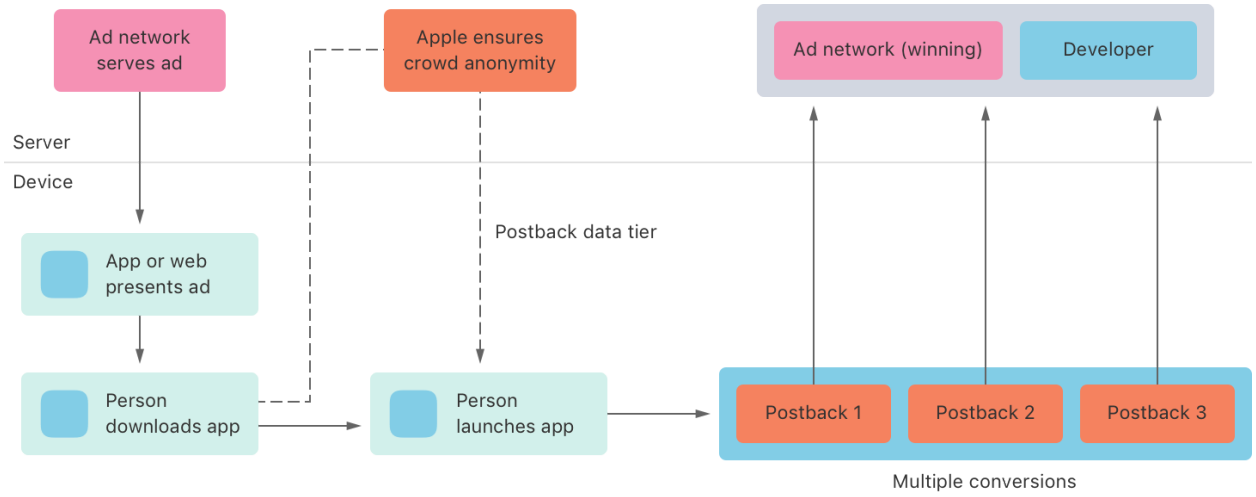
iOS Features

SKOverlay:



StoreKit:

SKAdNetwork:



Updates needed to Prebid Mobile:

- Needs to support passing of a bidRequest that includes SKAdNetwork ID and source app ID via BidRequest.imp.ext.skadn to the Prebid server.
- Support Bid Response that includes BidResponse.seatbid.bid.ext.skadn
- Support SKAdNetwork clicks / installs (SKOverlay, (Auto)StoreKit).
- Support attribution via SKAN from 1.0->4.0

Prerequisites for Integration:

- DSPs should register as an Ad Network to Apple's SKAdNetwork API. Click here to register.
- Supply your public key to Apple and set your postback urls.
- DSPs should be able to ingest the list of SKAdNetwork ID(s), version, and source bundle ID information in BidRequest.imp.ext.skadn object of the bid request.
- If the DSP is responding with a SKAdNetwork enabled campaign, it should respond with SKAdNetwork ID and all relevant fields in the bid response.
- The DSP has at least one SKAdNetwork ID registered in the publisher's info.plist
- The user's device is operating on iOS14 or above.
- The Application is integrated with Prebid Mobile SDK via a SKAdNetwork supported version (3.0.0).
- The DSP is on OpenRTB 2.4 and above spec

- ORTB SkAdNetwork Fields

No Ad Server

1. The Prebid SDK sends the bid request including SKAdNetwork ID and source app ID via BidRequest.imp.ext.skadn to the Prebid server.
2. Prebid server runs the header bidding auction among preconfigured demand partners.
3. Prebid Server responds with the winning bid (with SKAdNetwork attribution) that contains targeting keywords and click data/signature in BidResponse.seatbid.bid.ext.skadn
4. The rendering module renders the winning bid with click data signature.
5. User downloads and installs the app shown in the ad.
6. Install of the new app is registered to SKAdNetwork. Advertised app can pass a conversion value agreed by DSP and the advertisers (optional)
7. Clicks and installs are matched
8. Install data, conversion value (optional), and signature are sent via postback.
9. DSP attributes the install.

With Ad Server: Original API

1. The Prebid SDK sends the bid request including SKAdNetwork ID and source app ID via BidRequest.imp.ext.skadn to the 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 (with SKAdNetwork attribution) that contain targeting keywords and click data/signature in BidResponse.seatbid.bid.ext.skadn 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 of the winning bid.
6. The primary ad server responds with an ad. If the line item associated with the Prebid Mobile bid wins, the primary ad server returns the Prebid Universal Creative (PUC) to the ad server's SDK.
7. The primary ad server SDK starts the rendering received ad markup.
8. The PUC fetches creative content of the winning bid from the Previd Cache and renders it.
9. User downloads and installs the app shown in the ad.
10. Install of the new app is registered to SKAdNetwork. Advertised app can pass a conversion value agreed by DSP and the advertisers (optional)
11. Clicks and installs are matched
12. Install data, conversion value (optional), and signature are sent via postback.
13. DSP attributes the install.

With Ad Server: Rendering API

1. The rendering module sends the bid request including SKAdNetwork ID and source app ID via BidRequest.imp.ext.skadn to the Prebid server.
2. Prebid server runs the header bidding auction among preconfigured demand partners.
3. Prebid Server responds with the winning bid (with SKAdNetwork attribution) that contains targeting keywords and click data/signature in BidResponse.seatbid.bid.ext.skadn
4. The rendering module sets up the targeting keywords of the winning bid into the ad unit of 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.
9. User downloads and installs the app shown in the ad.
10. Install of the new app is registered to SKAdNetwork. Advertised app can pass a conversion value agreed by DSP and the advertisers (optional)
11. Clicks and installs are matched
12. Install data, conversion value (optional), and signature are sent via postback.
13. DSP attributes the install.

With Ad Server: Mediation API

1. The Prebid SDK sends the bid request including SKAdNetwork ID and source app ID via BidRequest.imp.ext.skadn to the Prebid server.
2. Prebid server runs the header bidding auction among preconfigured demand partners.
3. Prebid Server responds with the winning bid (with SKAdNetwork attribution) that contains targeting keywords and click data/signature in BidResponse.seatbid.bid.ext.skadn.
4. [OPTIONAL] The Prebid SDK sets up the targeting keywords of the winning bid into the ad unit of 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 a mediation chain.
7. The Primary Ad Server SDK runs the Waterfall.
8. If the mediation item contains the name Prebid Adapter it instantiates the respective class.
9. [OPTIONAL] adapters checks the whether the Line Item's targeting keywords match the bid targeting keywords
10. Adapter renders a winning bid cached in the SDK.
11. User downloads and installs the app shown in the ad.
12. Install of the new app is registered to SKAdNetwork. Advertised app can pass a conversion value agreed by DSP and the advertisers (optional)
13. Clicks and installs are matched
14. Install data, conversion value (optional), and signature are sent via postback.
15. DSP attributes the install.

Rendering & Delegation

The Prebid Mobile Community has been working closely with Teads to allow third party SDK's to be used for rendering. This will give publishers the option to write their own rendering SDK instead of using the Prebid Universal Creative, GAM, or another rendering solution. This solution will work across all ad formats and we will also be adding multiformat ad support with this effort as well.

The docs for this implementation are already available in the official Prebid Documentation:

Prebid Plugin Renderer (Android is complete, iOS is in progress. Both implementations will also be expanded to include all ad types)

Multi-format Ad API for iOS (will be expanded to include all ad types)

Multi-format Ad API for Android (will be expanded to include all ad types)