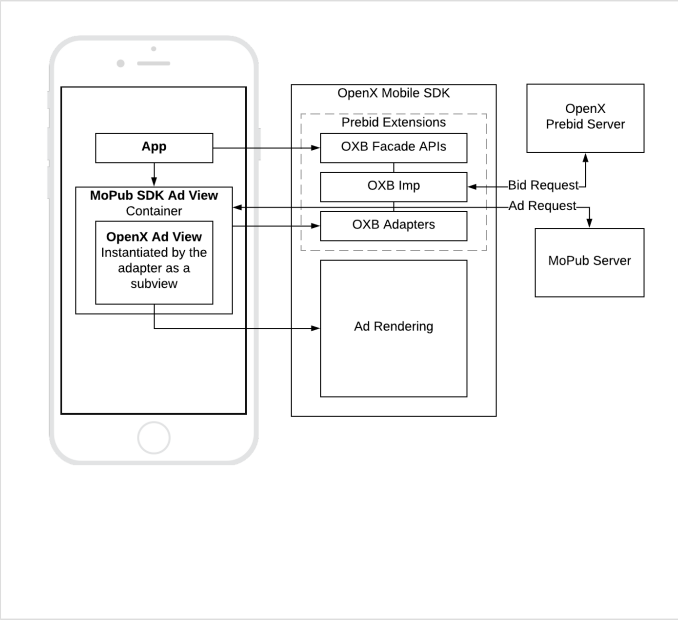
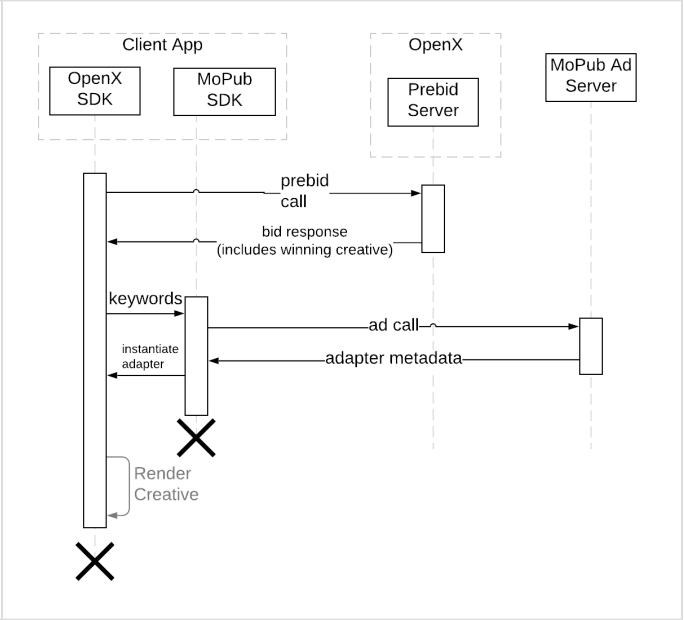
[EXTERNAL EXPORT] MoPub Facade

The support of MoPub as a Primary Ad Server is implemented using the mediation feature of MoPub SDK. MoPub's line item contains information about the mediation adapter. MoPub SDK instantiates the Prebid adapter which retrieves the winning creative from the cache and renders it.





Banner

MoPubBannerAdUnit - the object that serves as a bid requester which enriches the MoPub's ad view with targeting keywords. The publisher is able to customize the bid request properties. Also, the publisher should invoke the code for loading the ad explicitly in the callback of the bid request. To start the loading process the publisher have to explicitly invoke the **fetchDemand()** method. The usage is similar to the original Prebid Banner API.

Responsibilities:

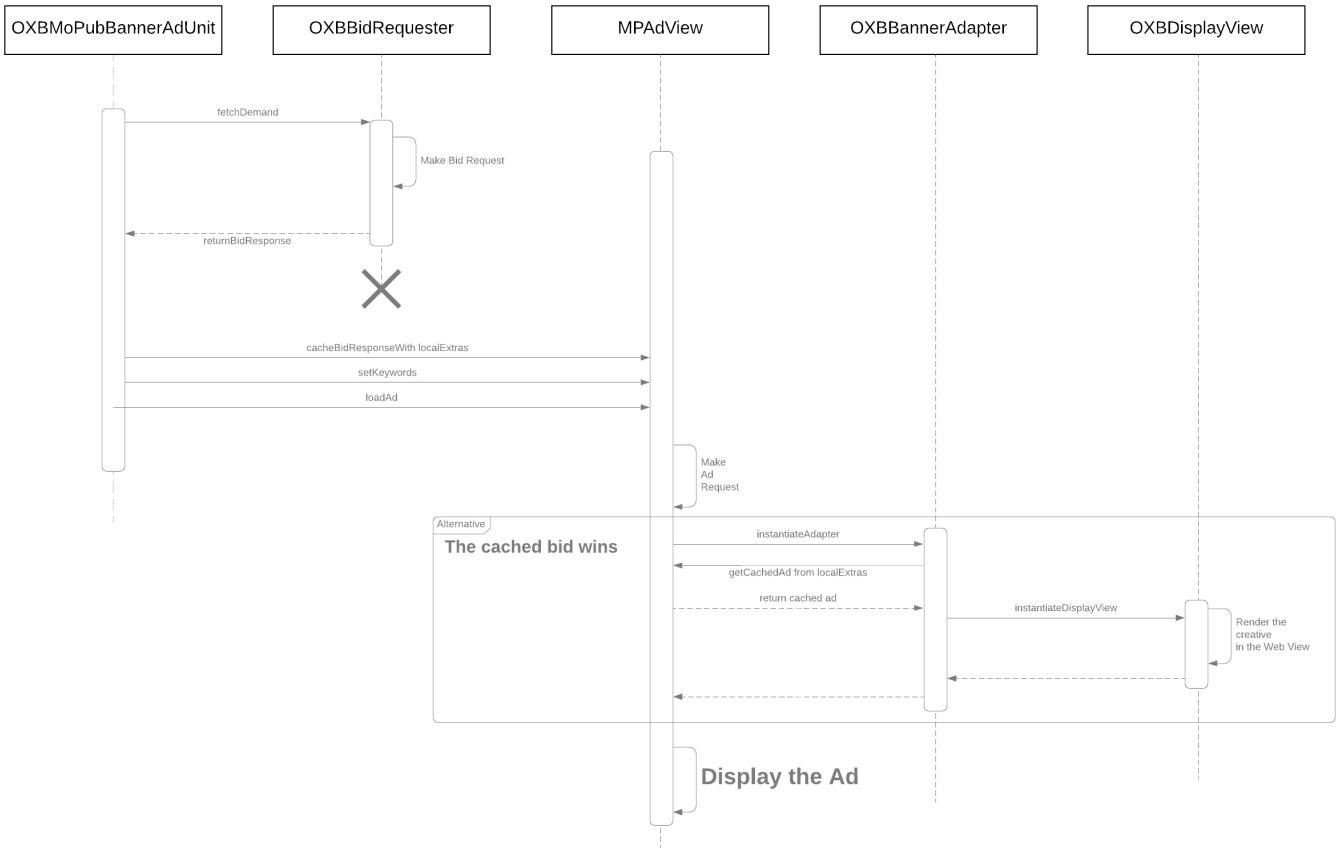
- Configure a bid request
- Make the bid request
- Enrich the MoPub ad view with targeting keywords.
- Invoke a callback with the status of the bidding.
- Repeat the bid request and reload the ad.

MoPubBannerAdapter - the classical MoPub's adapter for the custom network. It extracts the cached ad from the property *localExtras* and initiates the rendering of creative.

Responsibilities:

- Create a Prebid Ad View
- Configure the Prebid Ad View respectively to the bid response
- Delegate ad events to the MoPub engine

The sequence diagram for fetching the bids and displaying the ads with MoPub banner:



Native

NativeAdUnit - the object that serves as a bid requester which enriches the MPNativeAdRequester with targeting keywords. The publisher is able to customize the bid request properties. Also, the publisher should invoke the code for loading the MoPub ad explicitly in the callback of the bid request. To start the loading process the publisher has to invoke the **fetchDemand()** method explicitly.

Responsibilities:

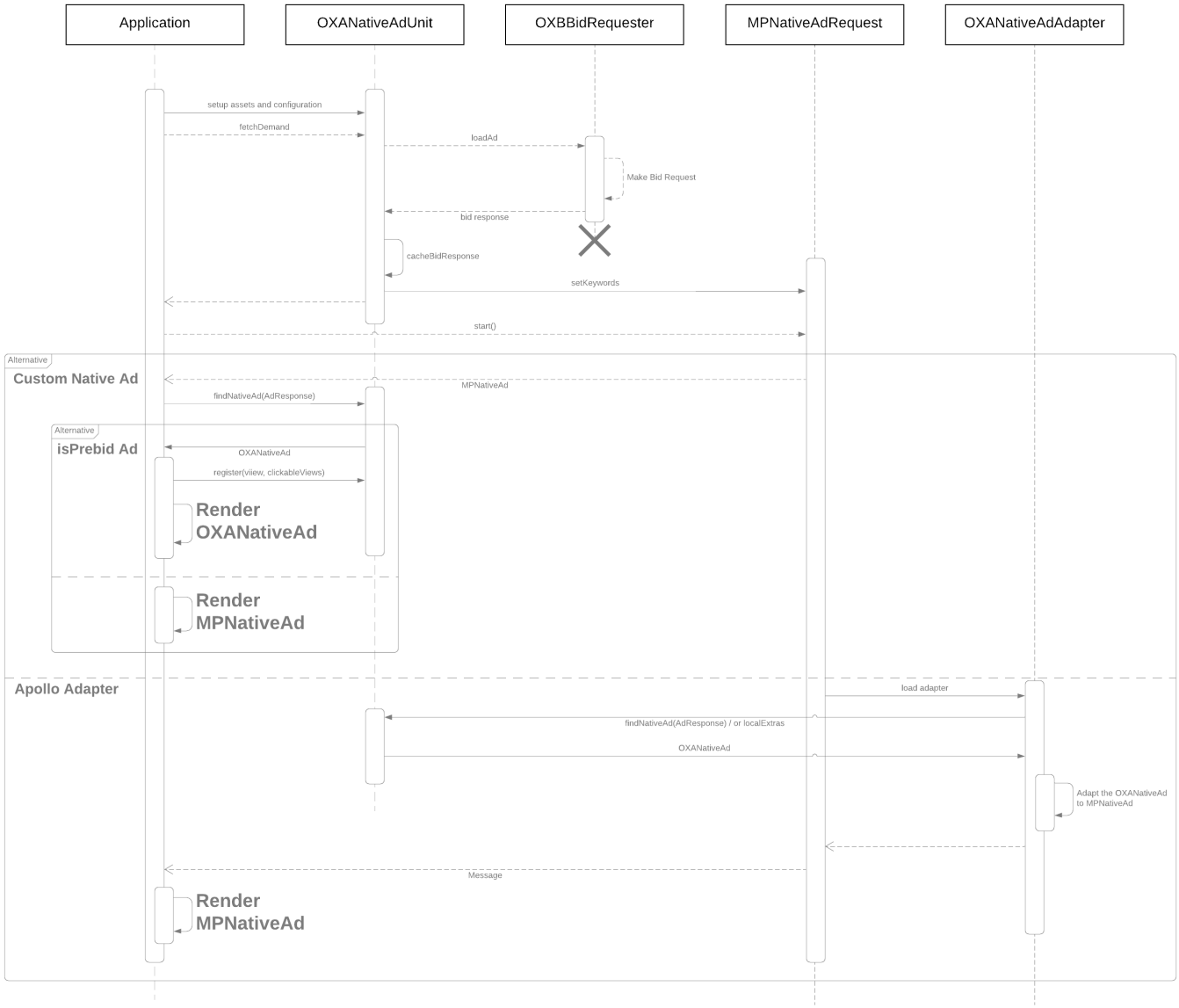
- Configure a bid request
- Make the bid request
- Enrich the MPNativeAdRequest with targeting keywords.
- Invoke a callback with the status of the bidding and NativeAd object if Apollo's ad source won.

NativeAd - the object that represents the native bid response. Publishers should use this object to initialize UI elements of the ad. In addition, the NativeAd is responsible for processing native event trackers, received in the bid response, impression trackers, and click trackers. So publishers should register the Ad View and clickable UI components in the NativeAd before adding the ad to the UI.

Responsibilities:

- Provide an access to the deserialized native bid response.
- Track native ad events like impression, click, viewability status, and others.

The sequence diagram for fetching the bids and displaying the ads combining the app's UI, MoPub Ad Loader, and NativeAdUnit:



Interstitial

MoPubInterstitialAdUnit - the object that serves as a bid requester which enriches the MoPub's interstitial controller with targeting keywords. The publisher is able to customize the bid request properties. Also, the publisher should invoke the code for loading the ad explicitly in the callback of the bid request. To start the loading process the publisher have to explicitly invoke the **fetchDemand()** method. The usage is similar to the Prebid Interstitial API.

Responsibilities:

- Configure a bid request
- Make the bid request
- Enrich the MoPub interstitial controller with targeting keywords.
- Invoke a callback with the status of the bidding.
- Repeat the bid request and reload the ad.

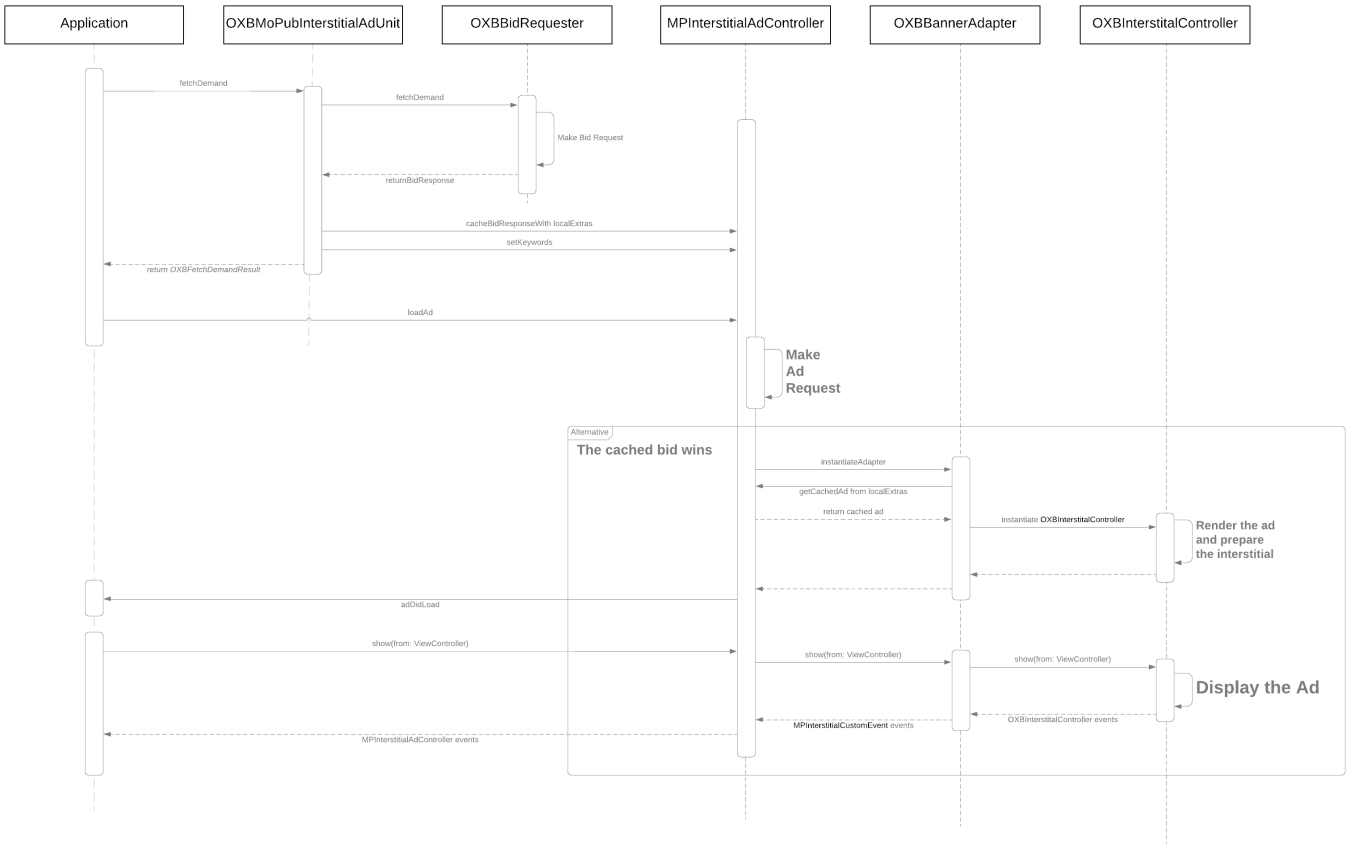
MoPubInterstitialAdapter - the classical MoPub's adapter for the custom network. It extracts the cached ad from the property *localExtras* and initiates the rendering of creative.

Responsibilities:

- Create an Interstitial Controller
- Configure the Interstitial Controller respectively to the bid response

- Delegate ad events to the MoPub engine

The sequence diagram for fetching the bids and displaying the ads with MoPub interstitial:



Rewarded

The API for Rewarded ads is based on the Interstitial API. The flow of loading and showing the ads is totally the same.

Since the rewarding functionality is performed by MoPub engine the only difference with Interstitial API is customizing the bid request with `is_rewarded_inventory` flag.

MoPubRewardedAdUnit - totally the same as `MoPubInterstitialAdUnit`.

MoPubRewardedAdAdapter - the classic adapter for MoPub's rewarded ad unit. It extracts the cached ad from the property *localExtras* and initiates the rendering of creative. The responsibilities of this adapter are the same as for the Interstitial adapter with one additional item - inform the MoPub engine that a user has earned a reward when the ad is completed.