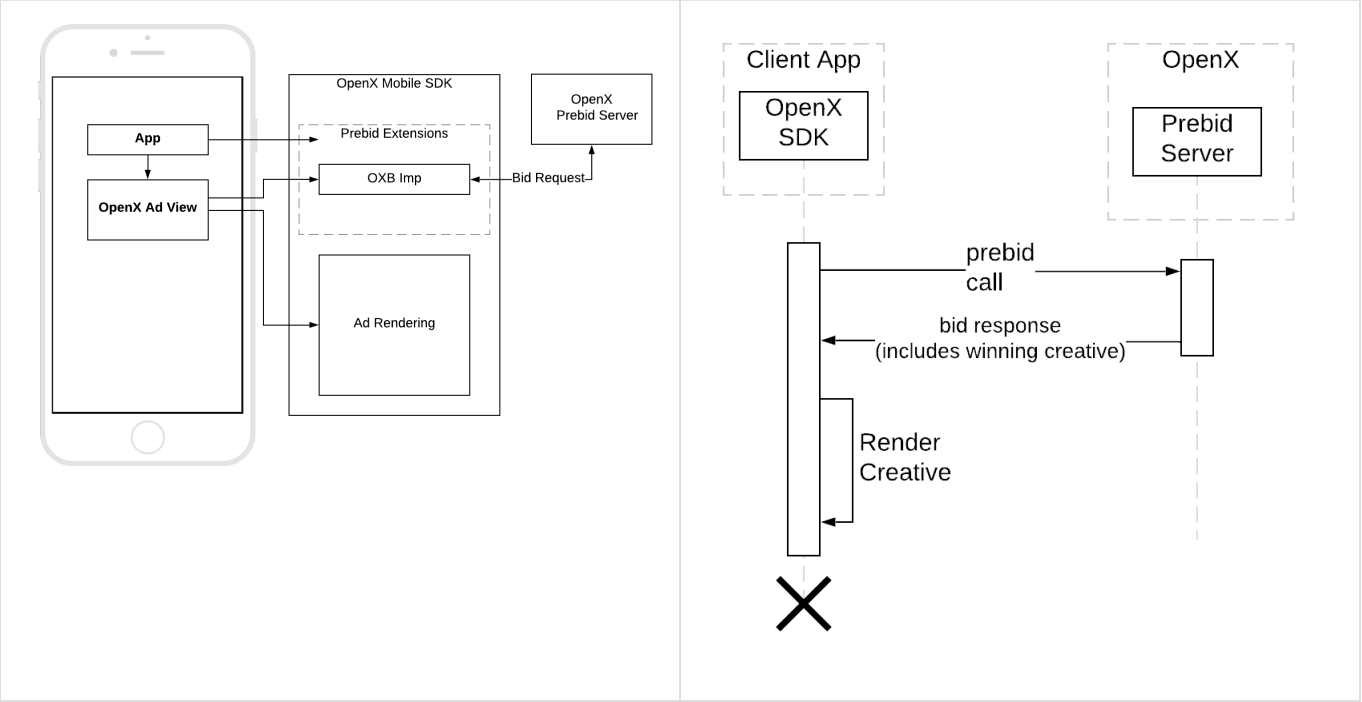
[EXTERNAL EXPORT] In-App Bidding Facade

The In-App Bidding integration works like a regular advertisement SDK. It makes bid requests to the prebid server. After receiving the winning bid SDK renders it in the UI.



Banner

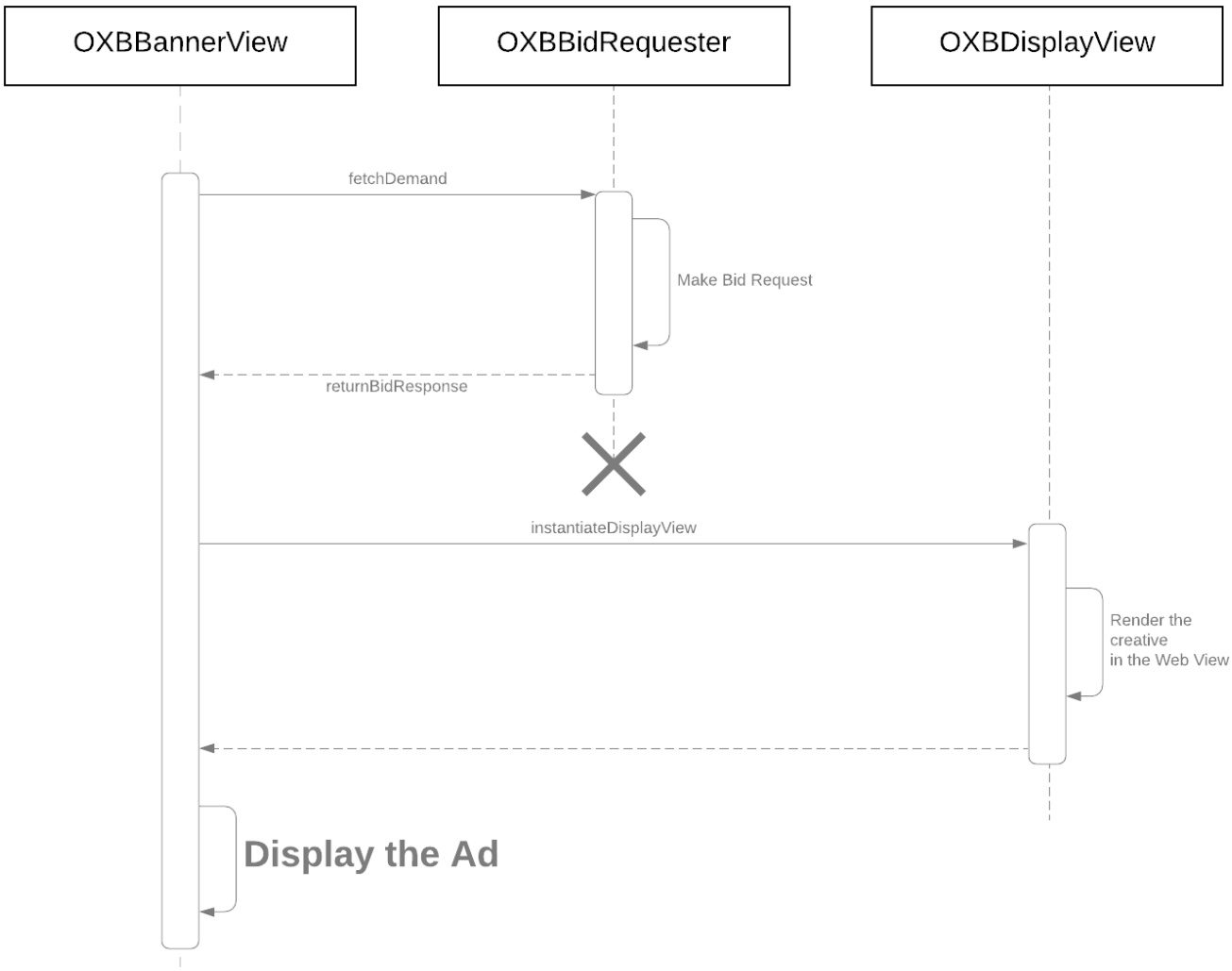
BannerView - the view object that shows the ads. The BannerView should be built into the UI like any other view. The publisher is able to customize the view properties and the bid request properties. To start the loading process the publisher have to explicitly invoke the load method.

Responsibilities:

- Configure a bid request
- Make the bid request
- Display the ad
- Inform the delegate about the flow stages

BannerViewDelegate - the interface that should implement the delegate class to handle events from BannerView.

The sequence diagram for fetching the bids and displaying the ads with In-App Bidding integration



Native

NativeAdUnit - the object that serves as a native ad requester. The publisher is able to customize the bid request properties To start the loading process the publisher has to explicitly invoke the **fetchDemand()** method.

Responsibilities:

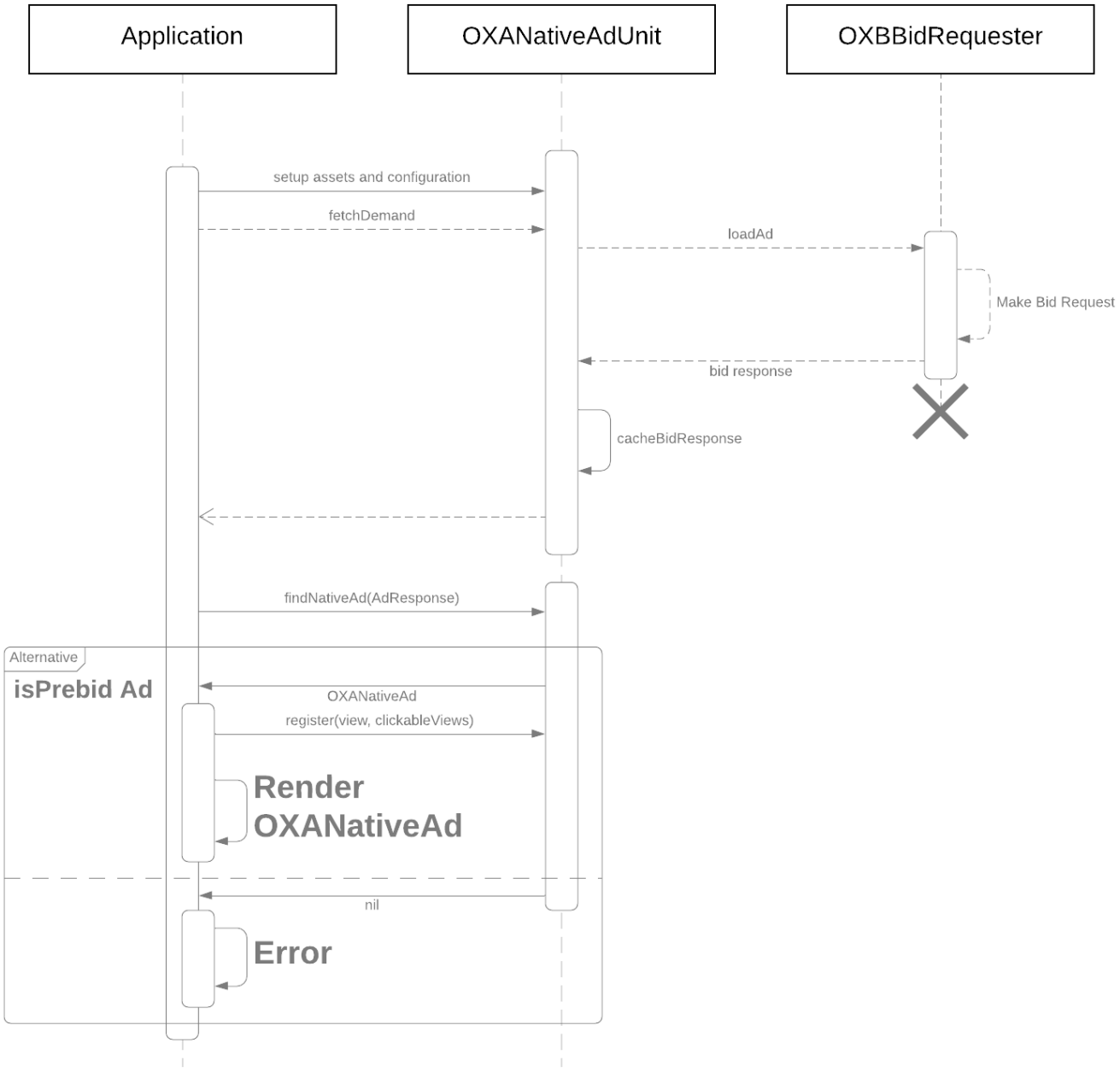
- Configure a bid request
- Make the bid request
- Invoke a callback with the status of the bidding and NativeAd object if any bid had won.

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, GAM Ad Loader, and NativeAdUnit:



Interstitial x

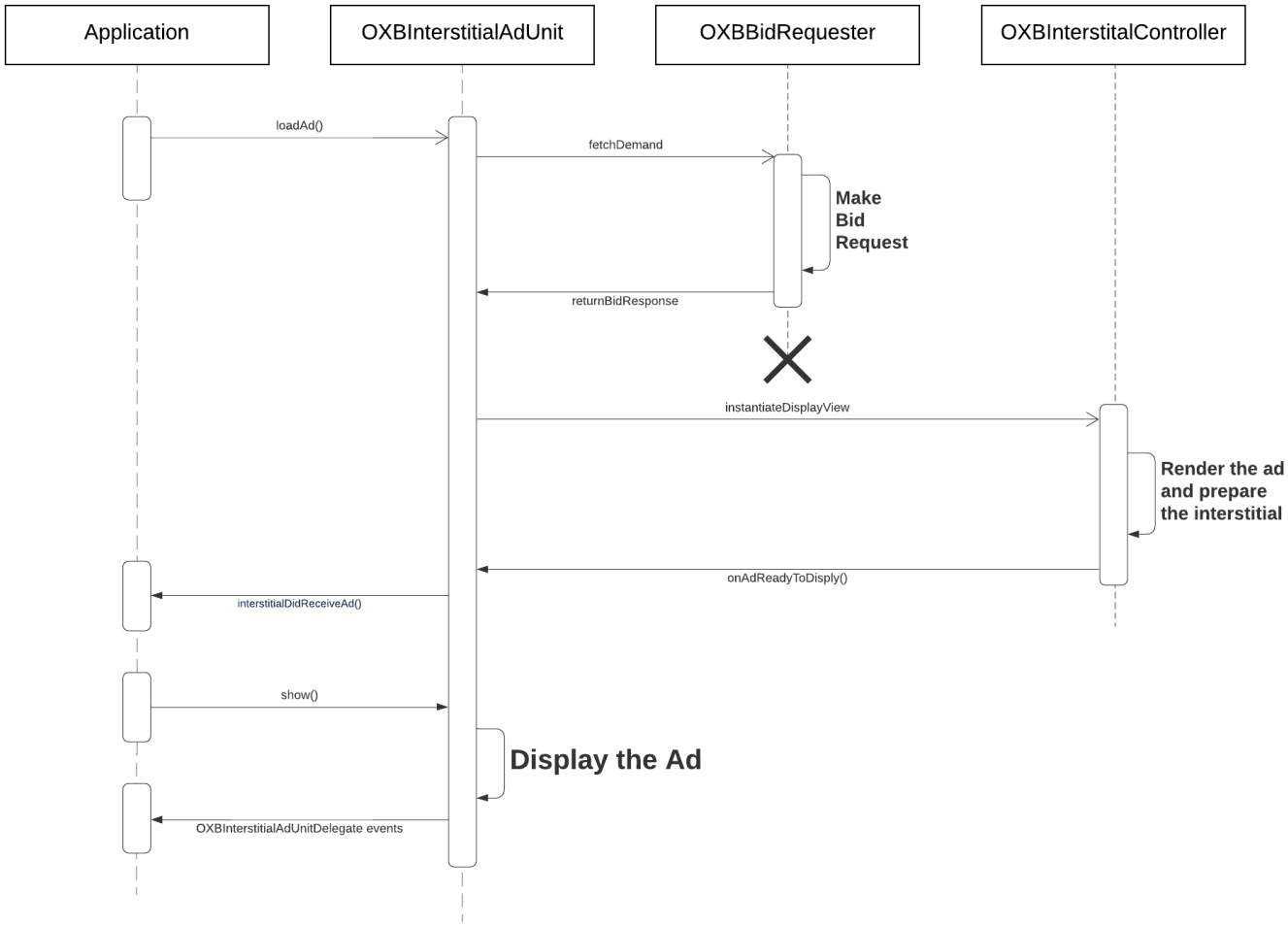
InterstitalAdUnit - the class that manages the interstitial ads. The InterstitalAdUnit should be instantiated in the app in the place (ViewConroller, Activity) where the interstitial ad should be presented. The publisher is able to customize the properties of the bid requests. To start the ad loading the publisher should invoke the loadAd() method, to show the interstitial - show().

Responsibilities:

- Configure a bid request
- Make the bid request
- Display the ad
- Inform the delegate about the flow's stages

InterstitalAdUnitDelegate - the interface that should implement delegate class to handle events from BannerView.

The sequence diagram for fetching the bids and displaying the ads with In-App Bidding integration:



Rewarded

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

The difference between those two ad formats is additional methods for rewarding the user at some point. So the `RewardedAdUnitDelegate` has an additional callback `userDidEarnReward` to inform the app that the reward has been received. The argument of this callback is `NSObject` on iOS and `Object` on Android. However, since the “reward” feature is not supported on the prebid UI the managing of the reward should be implemented on the publisher side.

RewardedAdUnit - totally the same as `InterstitialAdUnit`.

RewardedAdUnitDelegate - the same protocol as `InterstitialAdUnitDelegate`with *additional responsibility* - passing the Reward object to the delegate class in the app.