Proposed Design Document: Renderer Changes for Prebid SDK

Problem Statement

The current renderer design in the Prebid SDK faces the following challenges:

1. OS does not seem to support sending custom data in the bid request. However, the following method exists:

```
private PluginRenderer map(PrebidMobilePluginRenderer prebidMobilePluginRenderer) {
 PluginRenderer pluginRenderer = new PluginRenderer();
 pluginRenderer.setName(prebidMobilePluginRenderer.getName());
 pluginRenderer.setVersion(prebidMobilePluginRenderer.getVersion());
 pluginRenderer.setData(prebidMobilePluginRenderer.getData());
 return pluginRenderer;
}
```

Is there a way to leverage this or another method to enable sending custom data in iOS bid requests?

2. iOS appears to lack support for interstitial ads in the Prebid custom renderer. Are there plans to introduce this functionality, or is there a recommended approach for adding interstitial ad support?
   The current issue with interstitials lies in the InterstitialController. Specifically:

- The variable adViewManager (var adViewManager: PBMAdViewManager?) needs to be made public. Alternatively, should the PBMAdViewDelegate be implemented and honored by the custom renderer?
- Additionally, there is no way to receive a callback for showInterstitialAd in the customAdapter.Could you clarify or suggest the best approach to address these challenges?

3. The renderer maintains only a single object, which causes conflicts when multiple banner or interstitial objects are created, leading to overwriting.
4. The existing PrebidMobilePluginRenderer is not aligned with the complete implementation of the interstitial. It has no support for the show, isReady.

Proposed Solutions

1. Existing Architecture Solution

- Extend the current design to support interstitials by modifying the renderer's logic to handle multiple objects by Introducing an array or dictionary to store references for banners and interstitials, indexed by unique identifiers.
- Create the subclass of PBMAdViewManager and override the public methods. and assign

```
//
//  PrebidMobileCustomAdAdapter.swift
//  InternalTestApp
//
//  Created by Vikas Jangir on 22/11/24.
//  Copyright © 2024 Prebid. All rights reserved.
//

import Foundation
```

```swift
import UIKit
import PrebidMobile
import CustomAdSDK

public class PrebidMobileCustomAdAdapter: NSObject, PrebidMobilePluginRenderer {
    public let name = "PrebidMobileCustomAdAdapter"
    public let version = "10.8.0"
    private var banner: CustomADBanner?
    weak var bannerAdViewDelegate: PBMAdViewDelegate?

    var interstitialAdViewManager: CustomAdInterstitialAdViewManager?

    public var data: [AnyHashable: Any]? {
        guard let token = CustomADSdk.getToken() else {
            return nil
        }
        return ["token": token]
    }

    public func isSupportRendering(for format: AdFormat?) -> Bool {
        AdFormat.allCases.contains(where: { $0 == format })
    }

    public func setupBid(_ bid: Bid, adConfiguration: AdUnitConfig, connection: PrebidServerConnectionProtocol) {

    }

    public func createBannerAdView(with frame: CGRect, bid: Bid, adConfiguration: AdUnitConfig,
                                   connection: PrebidServerConnectionProtocol, adViewDelegate: (any PBMAdViewDelegate)?) {
        DispatchQueue.main.async {
            self.bannerAdViewDelegate = adViewDelegate
            _ = Data(base64Encoded: bid.adm ?? "")

            guard let adMarkup = bid.adm, let data = Data(base64Encoded: adMarkup) else {
                return
            }
            self.banner = CustomADBanner.init(frame: frame, placementId: Int64("1671208348087") ?? 0)
            self.banner?.delegate = self
            self.banner?.load(data)
        }
    }


    public func createInterstitialController(bid: Bid, adConfiguration: AdUnitConfig, connection: PrebidServerConnectionProtocol,
                                             adViewManagerDelegate adViewDelegate: InterstitialController?, videoControlsConfig: VideoControlsConfiguration?) {

        DispatchQueue.main.async { [weak self] in
            guard let self = self else { return }
            self.interstitialAdViewManager = self.createPBMAdViewManager(adConfiguration: adConfiguration, connection: connection, interstitialController: adViewDelegate)
            adViewDelegate?.adViewManager = self.interstitialAdViewManager
            self.interstitialAdViewManager?.createInterstitialAdObject(bid: bid, adConfiguration: adConfiguration)
        }
    }
}

extension PrebidMobileCustomAdAdapter {



    func createPBMAdViewManager(adConfiguration: AdUnitConfig, connection: PrebidServerConnectionProtocol, interstitialController: InterstitialController?) -> CustomAdInterstitialAdViewManager {
        let admanager = CustomAdInterstitialAdViewManager(connection: connection, modalManagerDelegate: nil)
        admanager.adViewManagerDelegate = interstitialController
        admanager.adConfiguration.isInterstitialAd = true
        admanager.adConfiguration.isOptIn = adConfiguration.adConfiguration.isOptIn
        return admanager
    }
}

extension PrebidMobileCustomAdAdapter: CustomADBannerDelegate {
    func banner(_ banner: CustomADBanner!, didReceiveWithMetaInfo metaInfo: [AnyHashable : Any]!) {
        print("Banner did receive ad")
    }

    public func banner(_ banner: CustomADBanner, didFailToReceiveWithError error: CustomADRequestStatus) {
        print("Banner did fail to receive ad with error: \(error.localizedDescription)")
    }

    public func bannerDidFinishLoading(_ banner: CustomADBanner) {
        print("Banner did finish loading")
        self.bannerAdViewDelegate?.displayView?().addSubview(banner)
        let adDetails = PBMAdDetails(rawResponse: "test", transactionId: "tedst")
        self.bannerAdViewDelegate?.adLoaded(adDetails)
    }

    public func banner(_ banner: CustomADBanner, didFailToLoadWithError error: CustomADRequestStatus) {
        print("Banner did fail to load with error: \(error.localizedDescription)")
        self.bannerAdViewDelegate?.failed(toLoad: error)
    }

    public func bannerAdCustomADpressed(_ banner: CustomADBanner) {
        print("Banner ad CustomADpressed")
        self.bannerAdViewDelegate?.adDidDisplay()
    }

    public func banner(_ banner: CustomADBanner, didInteractWithParams params: [String : Any]?) {
        self.bannerAdViewDelegate?.adWasClicked()
    }

    public func bannerWillDismissScreen(_ banner: CustomADBanner) {
    }

    public func bannerDidDismissScreen(_ banner: CustomADBanner) {
        self.bannerAdViewDelegate?.adDidCollapse()
    }

    public func bannerWillPresentScreen(_ banner: CustomADBanner) {
        self.bannerAdViewDelegate?.adDidExpand()
    }

    public func bannerDidPresentScreen(_ banner: CustomADBanner) {
    }

    public func userWillLeaveApplicationFromBanner(_ banner: CustomADBanner) {
        self.bannerAdViewDelegate?.adDidLeaveApp()
    }
}


```

```swift


class CustomAdInterstitialAdViewManager: PBMAdViewManager {
    var interstitial: CustomADInterstitial?

    func createInterstitialAdObject(bid: Bid, adConfiguration: AdUnitConfig) {
        guard let adMarkup = bid.adm, let data = Data(base64Encoded: adMarkup) else {
            let error = NSError(domain: "PrebidMobile", code: 0, userInfo: [NSLocalizedDescriptionKey: "Ad markup is missing"])
            self.adViewManagerDelegate?.failed(toLoad: error)
            return
        }

        self.interstitial = CustomADInterstitial(placementId: Int64("1671208348087") ?? 0)
        self.interstitial?.delegate = self
        self.interstitial?.load(data)
    }

    override func show() {
        print("CustomAdManager show")
        guard let viewController = self.adViewManagerDelegate?.viewControllerForModalPresentation() else {
            return
        }
        interstitial?.show(from: viewController)
    }

    override func isAbleToShowCurrentCreative() -> Bool {
        return interstitial?.isReady() ?? false
    }

    override func pause() {
        print("CustomAdManager pause")
    }
    override func resume() {
        print("CustomAdManager resume")
    }
    override func mute() {
        print("CustomAdManager mute")
    }
    override func unmute() {
        print("CustomAdManager unmute")
    }

    override func handleExternalTransaction(_ transaction: PBMTransaction) {
        print("CustomAdManager handleExternalTransaction")
    }
}
```

```swift


extension CustomAdInterstitialAdViewManager: CustomADInterstitialDelegate {

    //Notifies that the ad is received with Meta info.
    public func interstitial(_ interstitial: CustomADInterstitial, didReceiveWithMetaInfo metaInfo: 
CustomADAdMetaInfo) {
        print("DemoApp : Interstitial ad is received with meta info \(String(describing: metaInfo))")
    }

    public func interstitialDidFinishLoading(_ interstitial: CustomADInterstitial) {
        print("[DemoApp : \(#function)]")
        let adDetails = PBMAdDetails(rawResponse: "test", transactionId: "tedst")
        self.adViewManagerDelegate?.adLoaded(adDetails)
    }

    public func interstitialAdCustomADpressed(_ interstitial: CustomADInterstitial) {
        print("[DemoApp : \(#function)]")
        self.adViewManagerDelegate?.adDidDisplay()

    }

    public func interstitial(_ interstitial: CustomADInterstitial, didFailToLoadWithError error:
                      CustomADRequestStatus) {
        print("[DemoApp : \(#function)]")
        print("Interstitial ad failed to load with error \(error)")
        self.adViewManagerDelegate?.failed(toLoad: error)

    }

    public func interstitialWillPresent(_ interstitial: CustomADInterstitial) {
        print("[DemoApp : \(#function)]")
    }

    public func interstitialDidPresent(_ interstitial: CustomADInterstitial) {
        print("[DemoApp : \(#function)]")
        self.adViewManagerDelegate?.adDidExpand()
    }

    public func interstitial(_ interstitial: CustomADInterstitial, didFailToPresentWithError error:
                      CustomADRequestStatus) {
        print("[DemoApp : \(#function)]")
    }

    public func interstitialWillDismiss(_ interstitial: CustomADInterstitial) {
        print("[DemoApp : \(#function)]")
    }

    public func interstitialDidDismiss(_ interstitial: CustomADInterstitial) {
        print("[DemoApp : \(#function)]")
        self.adViewManagerDelegate?.adDidCollapse()
        self.adViewManagerDelegate?.adDidClose()
    }
    func interstitial(_ interstitial: CustomADInterstitial, didInteractWithParams params: [String : Any]?) {
        print("[DemoApp : \(#function)]")
        self.adViewManagerDelegate?.adWasClicked()
    }

    public func userWillLeaveApplicationFromInterstitial(_ interstitial: CustomADInterstitial){
        print("[DemoApp : \(#function)]")
        self.adViewManagerDelegate?.adDidLeaveApp()
    }
}

```

2. New Design Solution

Key Benefits of the New Design

1. **Support for Multiple Ads**:

The new design enables seamless management of multiple ad instances via unique identifiers, ensuring scalability and preventing overwrites.

2. **Improved Extensibility**:

Standardized APIs and modular architecture make it easy to add new ad formats or functionalities in the future.

3. **Granular Lifecycle Event Handling**:

The AdDelegate provides callbacks for key ad events like loading, presenting, dismissing, and clicking, giving publishers precise control and insights.

4. **Enhanced User Experience**:

Methods like isAdReady ensure ads are only displayed when fully prepared, avoiding disruptions and improving engagement.

5. **Decoupled and Flexible Design**:

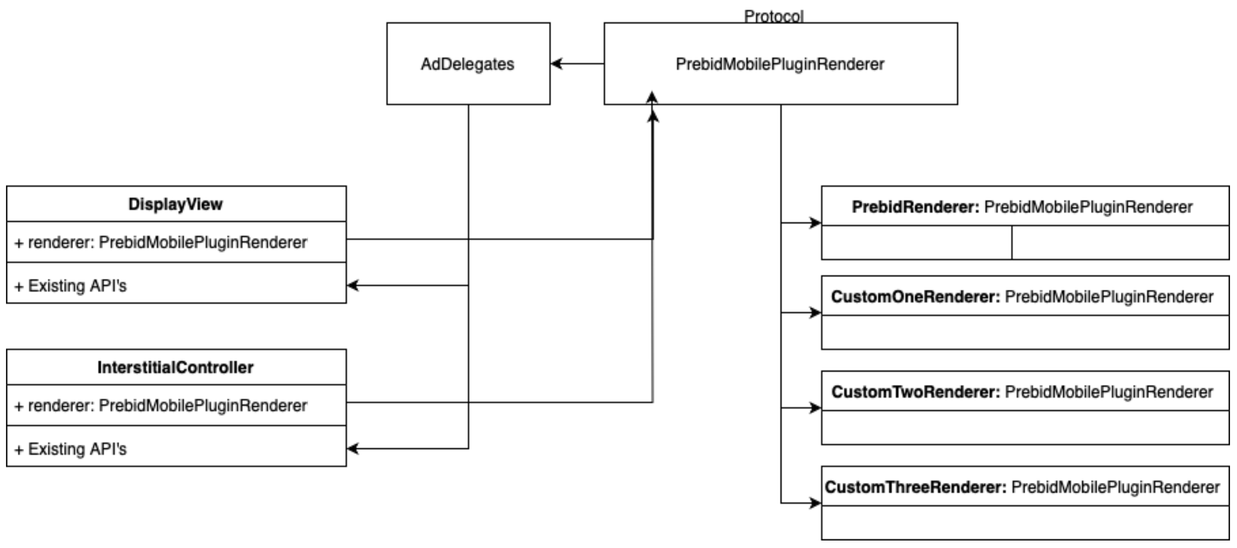
A delegate-based approach decouples renderer logic from publisher implementation, allowing easy customization and cleaner integrations.

```

import Foundation

@objc public protocol PrebidMobilePluginRenderer: AnyObject {

    /// Static property to retrieve the renderer's name
    @objc static var name: String { get }

    /// Static property to retrieve the renderer's version
    @objc static var version: String { get }

    /// Static property to retrieve additional data about the renderer
    @objc static var data: [AnyHashable: Any]? { get }

    /// Static method to check if a given ad format is supported by the plugin
    @objc static func isSupportRendering(for format: AdFormat?) -> Bool

    var delegate: AdDelegate? { get set}

    /// Creates and returns Banner View for a given Bid Response
    /// Returns nil in the case of an internal error
    @objc func createBannerAdView(with frame: CGRect, bid: Bid, adConfiguration: AdUnitConfig,
                                  connection: PrebidServerConnectionProtocol, adViewDelegate: (any
PBMAdViewDelegate)?)

```

```swift
    /// Creates and returns an implementation of PrebidMobileInterstitialControllerInterface for a given bid response
    /// Returns nil in the case of an internal error
    @objc optional func createInterstitialController(bid: Bid, adConfiguration: AdUnitConfig,
                                                     connection: PrebidServerConnectionProtocol,
                                                     adViewManagerDelegate: InterstitialController?,
                                                     videoControlsConfig: VideoControlsConfiguration?)

    /// Shows the ad if ready. Returns true if successful, false otherwise.
    @objc optional func showAd(for adUnitConfigFingerprint: String) -> Bool

    /// Checks if the ad is ready to be displayed
    @objc optional func isAdReady(for adUnitConfigFingerprint: String) -> Bool

    /// Register a listener related to a specific ad unit config fingerprint in order to dispatch specific ad events
    @objc optional func registerEventDelegate(pluginEventDelegate: PluginEventDelegate,
                                              adUnitConfigFingerprint: String)

    /// Unregister a listener related to a specific ad unit config fingerprint in order to dispatch specific ad events
    @objc optional func unregisterEventDelegate(pluginEventDelegate: PluginEventDelegate,
                                                adUnitConfigFingerprint: String)

    /// Setup a bid for a given ad unit configuration
    @objc func setupBid(_ bid: PrebidMobile.Bid, adConfiguration: PrebidMobile.AdUnitConfig,
                        connection: PrebidServerConnectionProtocol)

}


import Foundation

@objc public protocol AdDelegate: AnyObject {

    /// Called when the ad is successfully loaded
    @objc optional func adDidLoad(for renderer: PrebidMobilePluginRenderer)

    /// Called when the ad fails to load
    @objc optional func adDidFailToLoad(for renderer: PrebidMobilePluginRenderer, error: Error)

    /// Called when the ad will present
    @objc optional func adWillPresent(for renderer: PrebidMobilePluginRenderer)

    /// Called when the ad is presented
    @objc optional func adDidPresent(for renderer: PrebidMobilePluginRenderer)

    /// Called when the ad will be dismissed
    @objc optional func adWillDismiss(for renderer: PrebidMobilePluginRenderer)

    /// Called when the ad is dismissed
    @objc optional func adDidDismiss(for renderer: PrebidMobilePluginRenderer)

    /// Called when the user clicks on the ad
    @objc optional func adDidClick(for renderer: PrebidMobilePluginRenderer)

    /// Called when the user will leave the application due to the ad interaction
    @objc optional func adWillLeaveApplication(for renderer: PrebidMobilePluginRenderer)
```

```
}

```

Flow Diagram


Protocol
AdDelegates
PrebidMobilePluginRenderer
DisplayView
+ renderer: PrebidMobilePluginRenderer
+ Existing API's
InterstitialController
+ renderer: PrebidMobilePluginRenderer
+ Existing API's
PrebidRenderer: PrebidMobilePluginRenderer
CustomOneRenderer: PrebidMobilePluginRenderer
CustomTwoRenderer: PrebidMobilePluginRenderer
CustomThreeRenderer: PrebidMobilePluginRenderer