SAS® 360 Match

iOS SDK

Updated October 21, 2022

Contents

iOS SDK

The SAS® 360 Match IOS SDK provides UI elements encapsulating the request and rendering of SAS® 360 Match ads within iOS mobile applications. The API provides functionality similar to that of Apple's iAd Framework and Google's AdMob/DoubleClick API.

The SDK implementation supports version 2.0 of the IAB's MRAID specification for rich media advertisements. MRAID-compliant ads can request expansion to full screen or an arbitrary size, provide for two-part creatives, respond to visibility changes, control screen orientation, and engage with other device functions.

An ad placement is represented by either a `SASIA_Ad` or `SASIA_InterstitialAd`, both subclasses of `SASIA_AbstractAd`. The first type, `SASIA_Ad`, is used in cases where the ad is mixed with other application content in the same screen ("inline"). The second type, `SASIA_InterstitialAd`, is used in cases where the ad should appear full-screen, blocking other application content until the user explicitly dismisses it.

Each type is capable of rendering one ad at a time, although it can be reused and requested to show different ads over time. Each ad request is represented by a `SASIA_AdRequest` which encapsulates all targeting parameters for the ad request, including size. The same `SASIA_AdRequest` can be used multiple times, if appropriate, such as to refresh the ad object to show a different ad.

To show any type of ad, the application first creates the `SASIA_Ad` or `SASIA_InterstitialAd` object and asks it to load and render a `SASIA_AdRequest`. The request and rendering occur asynchronously. The application can provide the ad a `SASIA_AdDelegate` that receives callbacks when the ad has been loaded successfully (or not).

For inline ads, the `SASIA_Ad` provides access to its underlying view, so that the app can set its layout appropriately and include it in a UIViewController's view hierarchy. It is the app's responsibility to ensure the view size is appropriate for the size of ad requested from SAS® 360 Match. The app is also responsible for how and when the view appears (or not) while loading is in progress. Using conventional UI framework methods, the app can keep the view hidden until the load is successful, can show a superview containing an activity spinner, can show a placeholder image, etc.

For interstitial ads, the app merely calls the `SASIA_InterstitialAd`'s `showFromController` method. The ad is presented modally, blocking the application's regular user interface. A black "close" icon is automatically added to the upper right corner of the ad. MRAID-compliant ads can override showing the close icon to allow a creative-specific close graphic to appear.

By default, when a user touches an ad, its "click" action triggers a full-screen in-app interstitial view of the action's web page. A `SASIA_AdDelegate` can be notified when an in-app interstitial ad action has been dismissed by the user. If the app sets the ad's `actionInBrowser` property, the action launches a page in Safari instead and causes the app to be pushed to the background.

But the `SASIA_AdDelegate` can also block the normal ad action and take an action for itself by implementing the `willBeginAction` method. In combination with calling the ad's `executeJavaScript` method, this gives the app powerful methods to customize a user's interaction with the ad.

Setting Up the iOS SDK

The mobile SDK for iOS is accessible in a public github repository. The repository hosts versioning and manifest data that enable Swift Package Manager (SPM) to manage your project's dependency on the SDK framework.

If you previously used directly downloaded packages for the mobile SDK, remove any references to the previous SDK framework from your project before you set up the SDK using SPM.

To use SPM to add the repository to your Xcode project, add a package dependency in your Xcode project. The URL for the mobile SDK for iOS is https://github.com/sassoftware/ci360-mobile-ios-sdk. When you are finished, verify that the SASCollector package is successfully added. For more information about adding a Swift package to an iOS project, see the Swift Package Manager documentation.

Compatibility

The SDK framework is built with "iOS Deployment Target" set to "iOS 6.0", and so is compatible with devices running iOS 6.0 and higher.

The SDK provides view containers that support rendering MRAID 2.0-compliant ads. All MRAID functions and behaviors are supported as described in the IAB specification "Mobile Rich-media Ad Interface Definitions (MRAID) v.2.0", dated April 16, 2013, with these clarifications and exceptions:

- Knowledge of the current visibility of an inline (non-interstitial) ad requires assistance from the app—specifically, the UIViewController hosting the ad. Unless the app provides such support, an inline ad is always regarded as "visible". See the SASIA_AbstractAd's `didChangeVisibility` method below. Interstitial ads take care of this automatically.

- An inline ad in the "default" state takes no action itself when MRAID `close()` is called. A callback is provided so the app can take an appropriate action (e.g. hiding the ad, loading a new ad, or ignoring the close altogether). See the SASIA_AdDelegate's `willClose` method below.

- The "storePicture" and "createCalendarEvent" MRAID methods are not yet supported.

Note that the SDK uses a custom NSURLProtocol to intercept an ad's request to load mraid.js in a script source tag. If the app also needs to create custom NSURLProtocols, those protocols should either be registered before the first ad is rendered, or should be coded to explicitly not handle any resource ending with "mraid.js".

Building

The “SASIA_SDK.framework” file contains several static library versions of the SDK to accommodate 32- and 64-bit apps on both devices and simulators. This file must be added to the app’s Frameworks.

Include files must be referenced from app source code as, e.g.:

```
#import <SASIA_SDK/SASIA_Ad.h><uses-permission
```

The SDK also includes a bundle file, “SASIA_SDK.bundle”, containing several necessary resources. Add this file to the app’s set of Supporting Files.

Changes from Version 2.x

- MRAID 2.0 ads are now supported, rather than just MRAID 1.0.
- iOS 6.0 is the minimum version supported.
- Static methods in SASIA_AdRequest must now be used to set the SASIA customer id and domain used by subsequent ad requests. Prior versions relied on supplying these with each initialization of a SASIA_AdRequest.
- The SASIA_Ad class is now a subclass of UIView, and therefore can be referenced when building UI layouts with Interface Builder.

Classes

SASIA_AbstractAd

`@property (nonatomic, weak) id<SASIA_AdDelegate> delegate`

- The delegate for the ad. Defaults to nil (no delegate).
- The SASIA_AbstractAd only maintains a weak reference to this object. The app should maintain a strong reference while the delegate is assigned to this ad.

`@property (nonatomic, readonly) UIViewController *viewController`

- The controller of the view for this ad. For a `SASIA_Ad`, this is the same `UIViewController` given to its initializer. For a `SASIA_InterstitialAd`, this is the controller of the interstitial view of the ad.

`@property (nonatomic, readonly) UIWebView *webView`

- The underlying UIWebView that renders the ad. This view is a subview of the top-level ad view and should not be added to the app’s view hierarchy directly. Access is provided here in cases where the app needs to further customize settings in the UIWebView.

`- (void) load:(SASIA_AdRequest *)adRequest`

- Initiates asynchronous loading and rendering of a new ad in the ad's view.
- Upon successfully loading a non-default ad, the delegate's `didLoad:` method is called.
- Upon successfully loading a default ad, the delegate's `didLoadDefault:` method is called.
- Upon failure, the delegate's `didFailLoad:error:failingUrl:` method is called.

`@property (nonatomic, readonly) NSInteger fcid`

- The internal SAS® 360 Match ID of the flight creative (ad) loaded. Note the FCID is only known for ads set to use beacon counting.
- =-2 when an ad with an unknown FCID is loaded.
- = -1 when no ad is loaded.
- = 0 when a default is loaded.
- >0 when an ad with a known FCID is loaded.

`@property (nonatomic, readonly, getter=isLoaded) BOOL loaded`

- YES when a non-default ad is successfully loaded.

`@property (nonatomic, readonly, getter=isDefaultLoaded) BOOL defaultLoaded`

- YES when a default ad is successfully loaded.

`- (void) close`

- Requests closing presentation of this ad. The SASIA_AdDelegate's `willClose` and `didClose` methods are called and have the effect as described in the SASIA_AdDelegate API section below.
- This method has the same effect as an MRAID-compliant ad calling the MRAID `close()` method.
- For a `SASIA_InterstitialAd`, this method is equivalent to the user touching the ad's close icon.

```
@property (nonatomic, readonly, getter=isActionInProgress) BOOL
actionInProgress
```

- YES when the ad has switched to its in-app “action” mode, wherein another view is temporarily presented on top of the regular application user interface, in response to the user touching this ad, and for the purpose of further engagement with the ad or advertiser.
  This property is not set when an ad action is launched in the device’s browser (`actionInBrowser` is YES).

```
@property (nonatomic, assign, getter=isActionInBrowser) BOOL actionInBrowser
```

- When NO (the default), an ad’s action will be presented within the app as a full-screen interstitial.

- When YES, the action will be presented in Safari, and the app is sent to the background.

```
@property (nonatomic, readonly) UIViewController *actionViewController
```

- The controller associated with display of the ad’s action view. This is non-nil only when a user has touched an ad and caused its action view to be presented within the app.

```
@property (nonatomic, assign) UIModalTransitionStyle actionTransitionStyle;
```

- The animation style to use when animating the appearance of this ad’s “action” view, which appears as a full-screen interstitial. The default is `UIModalTransitionStyleCoverVertical`.

```
- (void) cancelAction
```

- Cancels any in-app action initiated from this ad. If such an action is in progress, the action’s view is removed, allowing the application’s user interface to become active again. If an ad action is launched in Safari (`adActionInBrowser` is YES), this method has no effect.

```
- (void) didChangeVisibility:(BOOL)visible
```

- Informs the ad of changes to its visibility, typically determined by the UIViewController hosting it. This method exists solely to allow informing MRAID-compliant ads of visibility changes, so that they may adjust their behavior when becoming visible or not visible. This happens automatically for a `SASIA_InterstitialAd`, but a `SASIA_Ad` needs the assistance of its parent UIViewController.

```
- (NSString *) executeJavaScript:(NSString *)js jsStringExpression:(NSString
    *)jsStringExpression
```

- Executes arbitrary JavaScript in the underlying UIWebView holding the most recently loaded ad. This method can be used to inspect the ad's contents, manipulate it, etc.

- The js string, if non-null, must consist of complete JavaScript statements, functions, etc. This JavaScript is executed immediately before evaluating the jsStringExpression.

- The jsStringExpression, if non-null, must be a JavaScript expression yielding a string. The result of evaluating this expression is returned as the method result. If nil, a nil string is returned.

- The JavaScript is executed synchronously. If execution of js or jsStringExpression takes more than 10 seconds, execution is cancelled and, in the case of jsStringExpression, a nil is returned. If either of the strings provided are not valid JavaScript, it is not executed and, in the case of jsStringExpression, a nil is returned.

SASIA_Ad (extends SASIA_AbstractAd)

```
- (id) initForController:(UIViewController *)hostViewController
    withFrame:(CGRect)frame
```

- Initializer.

- `hostViewController` is the app's `UIViewController` that will include this ad's view in its view hierarchy.

- `frame` is the frame to be applied to the ad's view.

```
@property (atomic, readonly) UIView *view
```

- Returns the view hosting presentation of the ad. This method is provided for compatibility with previous versions, as the SASIA_Ad is itself the UIView hosting the ad.

- This view should be added to the app's `UIViewController`'s view hierarchy, as necessary, to mix this ad's display with other app content on the same screen. Loading of an ad into this view (via `load`) can be done before or after the view has been added to the `UIViewController`'s view hierarchy.

SASIA_InterstitialAd (extends SASIA_AbstractAd)

```
@property (nonatomic, assign) UIModalTransitionStyle
interstitialTransitionStyle
```

- The animation style to use when animating the appearance of this ad (via `showFromController`). The default is `UIModalTransitionStyleCoverVertical`.

```
- (id) init
```

- Initializer.

```
public void showFromController:(UIViewController *)hostViewController
```

- Presents this ad as a full-screen interstitial covering the regular application user interface. Loading of an ad (via `load`) can be done before or after this method is called.

- `hostViewController` is the app’s `UIViewController` requesting showing this `InterstitialAd`.

SASIA_AdRequest

```
+ (void) setDomain:(NSString *)domain;
+ (NSString *) domain;
```

- Setter and getter for the ad serving domain used in all subsequently-created requests. If the domain doesn't start with "http://" or "https://", "http://" is assumed.

- This method must be called once before instantiating any SASIA_AdRequest objects.

```
+ (void) setCustomerId:(NSString *)customerId;
+ (NSString *) customerId;
```

- Setter and getter for the customer's SAS® 360 Match ID used in all requests that are created subsequently.

- This method must be called once before instantiating any SASIA_AdRequest objects.

```
- (id) initWithTags:(NSDictionary *)tags
```

- Initializer for data encapsulating an ad request.

- `tags` is the complete collection of tags and values to supply in the ad request, represented as key/value pairs. The tag names and their values should be the same would be required for conventional SAS® 360 Match ad serving.

- For tags without values (such as NOCOMPANION), supply an `NSNull` for the value (i.e. `[NSNull null]`).

```
- (id) initWithURL:(NSString *)url
```

- Initializer that provides the complete URL to be used in the request, including the domain and customer id.

Note that the older initialization method, `initWithDomain:customerId:tags:`, is considered deprecated.

SASIA_MRAIDWebView

Each SASIA_Ad and SASIA_InterstitialAd creates a SASIA_MRAIDWebView, which is a container for holding the underlying ad content and which provides the additional functionality required by MRAID-compliant rich media ads. There is no need for an app to instantiate SASIA_MRAIDWebView objects directly.

But an app may set any of several static features of SASIA_MRAIDWebView, as desired, described below.

```
+ (void) setMraidTracing:(BOOL)trace;
+ (BOOL) mraidTracing;
```

- Setter and getter for a tracing option, used for troubleshooting MRAID ads. The default is NO. When YES, the MRAID behavior of loaded ads is traced in detail in the XCode console.

```
+ (void) supportSMSText:(BOOL)support;
+ (BOOL) supportsSMSText;
```

- Setter and getter for whether an MRAID ad should be allowed to send an SMS text message.

- The default is YES.

```
+ (void) supportTelephone:(BOOL)support;
+ (BOOL) supportsTelephone;
```

- Setter and getter for whether or not an MRAID ad should be allowed to initiate a telephone call.

- The default is YES.

```
+ (void) supportPicture:(BOOL)support;
+ (BOOL) supportsPicture;
```

Setter and getter for whether or not an MRAID ad should be allowed to store a picture on the device.
The default is NO.
NOTE: The picture storage feature is not yet supported by SASIA_MRAIDWebView, and so "supportsPicture" always returns NO.

```
+ (void) supportCalendar:(BOOL)support;
+ (BOOL) supportsCalendar;
```

- Setter and getter for whether or not an MRAID ad should be allowed to store a calendar event on the device.

- The default is NO.

- NOTE: The calendar event storage feature is not yet supported by SASIA_MRAIDWebView, and so "supportsCalendar" always returns NO.

SASIA_AdDelegate

To receive callbacks, an app must create an object implementing the SASIA_AdDelegate protocol, implementing whichever methods are of interest.

`- (void) didLoad:(SASIA_AbstractAd *)ad`

- Called after an ad succeeds loading an advertisement (initiated by `load`), and the ad loaded is not a default.
- Typically, this should trigger adding the ad view to the user interface, or animating it into position, etc.
- `ad` is the ad just loaded.

`- (void) didLoadDefault:(SASIA_AbstractAd *)ad`

- Called after an ad succeeds loading an advertisement (initiated by `load`), but the ad loaded is a "default" ad. Receipt of a default ad means that the ad server currently had no other ad to serve that could fulfill the `SASIA_AdRequest` made.
- `ad` is the ad just loaded.

```
- (void) didFailLoad:(SASIA_AbstractAd *)ad error:(NSError *)error
    failingUrl:(NSString *)url
```

- Called after an ad fails loading an advertisement (initiated by `load`).
- `ad` is the ad that failed to load.
- `error` describes the error.
- `failingUrl` is the URL of the resource that could not be loaded.

`- (BOOL) willClose:(SASIA_AbstractAd *)ad`

- Called when an ad is about to close. This occurs when a user touches a `SASIA_InterstitialAd`'s close icon, or when an MRAID-compliant ad asks to be closed, or when the ad's `close` method is called.
- `ad` is the ad that is about to close.
- Returning YES for a `SASIA_InterstitialAd` allows it to remove itself from the screen. Returning NO blocks it from closing.
- A `SASIA_Ad` takes no action itself upon close, but this method allows the ad's UIViewController to take some action to close the ad, if it desires. Returning YES only changes the ad's MRAID state to "hidden", and returning NO leaves the state alone.

`- (void) didClose:(SASIA_AbstractAd *)ad`

- Called when an ad is closed.

- `ad` is the ad that was closed.

`- (BOOL) willBeginAction:(SASIA_AbstractAd *)ad url:(NSString *)url`

- Called when an ad is about to initiate an action in response to a user touching some portion of the ad, or in response to an MRAID-compliant ad calling the `open()` method.
- `ad` is the ad whose action is about to begin.
- `url` is the destination URL for the action.
- Returning NO blocks the action from occurring.

`- (void) didFinishAction:(SASIA_AbstractAd *)ad`

- Called when an ad action's interstitial view is dismissed. When the ad action was shown in Safari, this method is not called.
- `ad` is the ad whose action has just finished.

`- (BOOL) willExpand:(SASIA_AbstractAd *)ad url:(NSString *)url`

- Called when an MRAID-compliant ad is about to expand itself to cover the current screen.
- `ad` is the ad that will expand.
- `url` is null if the ad will simply render itself in the expanded area. If non-null, then it is the destination URL of additional content that will be displayed in the expanded area.
- Returning NO blocks the expansion from occurring.

`- (void) didFinishExpand:(SASIA_AbstractAd *)ad`

- Called when an ad's expanded display has closed.
- `ad` is the ad whose expanded display has just closed.

`- (BOOL) willResize:(SASIA_AbstractAd *)ad size:(CGRect)size`

- Called when an MRAID-compliant ad is about to resize itself and break out of its current screen.
- `ad` is the ad that will resize.
- `size` gives the position and size the ad will adopt after being resized.

- Returning NO blocks the resize from occurring.

```
- (void) didFinishResize:(SASIA_AbstractAd *)ad
```

- Called when an ad’s resized display has closed and the ad has returned to its former position and size.
- `ad` is the ad whose resized display has just closed.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies. Copyright © 2022, SAS Institute Inc. All rights reserved.