Datami iOS SDK

Introduction

Datami SDK (aka SmiSdk) for iOS provides libraries, code samples and documentation to help developers building Sponsored Data enabled applications. It supports the following types of use-cases:

a) Sponsoring entire application - all HTTP(s) traffic from the App using the standard device frameworks will be enabled for sponsorship.

b) Sponsoring specific content URL – a single HTTP(S) URL may be sponsored in the App instead of the entire App

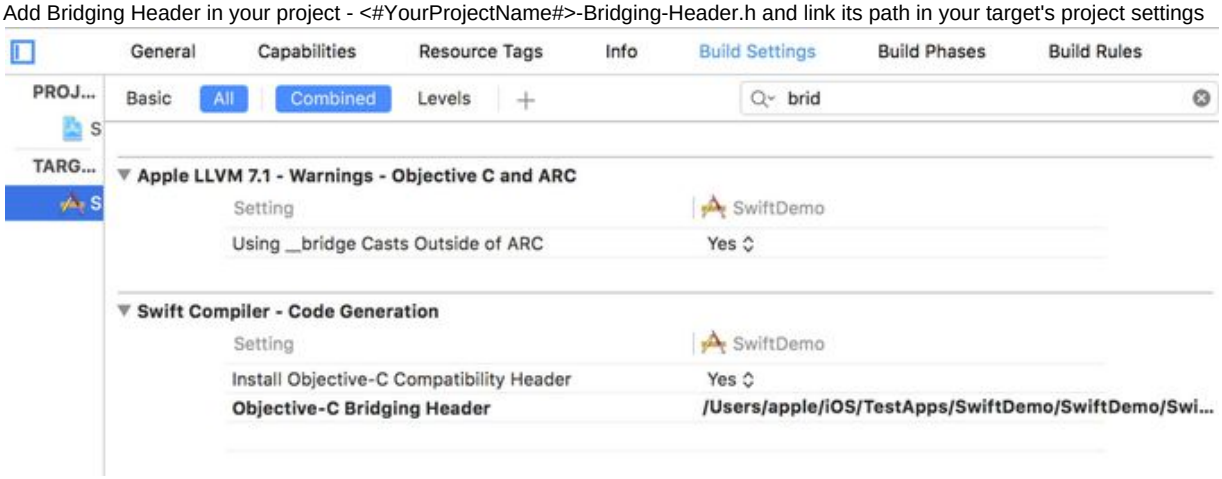
Assumptions

1. An SDK API access key is provisioned in Datami and provided to the application for integration with Datami SDK.
2. A sponsored data campaign is configured in the Datami portal for testing the application with Datami SDK.
3. Application developer has a physical iOS device with a SIM that works on the mobile network which is integrated with Datami.

Integration Steps

Datami will provide an **API key** required for the integration. Use the **API key** to activate the SDK through the API.

- **For complete App**
  1. Include the Datami iOS SDK library (libsmisdk.a) and SmiSdk.h in your project.
  2. Include required frameworks (See 'Required Frameworks' section below) in your projects
  3. Call 'initSponsoredData' api in your application at the start of your AppDelegate's 'didFinishLaunchingWithOptions' method (See Section 1.1). Note: The SDK must be initialized before App initiates any network traffic so the user does not get charged for the cellular data.
  4. Setup Listener to receive sponsored status from Datami SDK (See Section 1.2).
  5. Setup User Communication (See Section 1.3). This is optional and should be done to adequately communicate the sponsored data status to the user.
  6. Use SmiUIWebView for covering videos in UIWebView (See Section 1.4)
  7. Add Datami domain as exception domain in your application plist (See Section 3)
  8. Add custom configuration with 'registerAppConfiguration' for default NSURLSessionConfiguration.
- **For Single Content URL**
  1. Include the Datami iOS SDK library (libsmisdk.a) and SmiSdk.h in your project.
  2. Include required frameworks (See 'Required Frameworks' section below) in your projects.
  3. Call the api getSDAuth(See section 2.1) for any url that needs to sponsored.
- **Integration in swift**
  1. Include the Datami iOS SDK library (libsmisdk.a) and SmiSdk.h in your project.
  2. Include required frameworks (See 'Required Frameworks' section below) in your projects.
  3. Add Bridging Header in your project - <#YourProjectName#>-Bridging-Header.h and link its path in your target's project settings



  4. Add Class to Bridging-Header

     In YourProject-Bridging-Header.h:

     #import "SmiSdk.h"
  5. add **-ObjC** in Build Settings -> Linking -> Other Linker Flags

Required Frameworks

Framework	Comments
SystemConfiguration.framework	
CoreTelephony.framework	

Compatibility

The Datami iOS SDK is compatible with iOS 8 or higher.

Limitations

1. No support for UDP or RTP/RTSP in the current release.
2. If App makes any direct socket connections or VPN connections with the back-end – it requires a custom API. Not supported by standard SDK.
3. If the App has embedded video that is hosted by third parties and uses the third party video players or third party java scripts to play the videos - this may require Datami custom API for third party videos and varies case to case depending on the implementation. Datami can recommend the best approach for this use case after looking at the implementation.
   NOTE: if the app uses native player then the videos can be supported using the URL api already available in the SDK.
4. Both Facebook and Youtube videos are not supported in the current release.
5. Use cases that are supported with additional integration:
   a. Videos that use native player
   b. Some third party video players.

API Reference

I. API for sponsoring entire application

Datami iOS SDK includes an API to facilitate sponsored data solution for entire application. This api needs be only called once during application startup. The API method signature is as follows:

Note: getAppSDAuth API is deprecated, please use initSponsoredData API instead.

+(void)initSponsoredData:(NSString*)sdkKey userId:(NSString*)userId showSDMessage: (Boolean) showSDMessage exclusionDomains:(NSArray*) domains;

initSponsoredData API parameter details:

Name	Type	Description	[M]andatory/[O]ptional
sdkKey	NSString	The API Access Key provided by Datami	M
userId	NSString	The userId of the application. When provided, it can be used for analytics and/or authorization purposes. The userId can be email, username, phonenumber etc. Datami uses an internal id when userId is not provided	O
showSDMessage	Boolean	'YES' for opting SDK driven user messaging	M
exclusionDomains	NSArray	Array of exclusion domains (ex. www.example.com, www.google.com, etc.) that need to be excluded for sponsored data. Notes: 1. Sub-domains are not excluded by default. Any sub-domain that needs to be excluded should be included in the parameter 'exclusionDomains'. 2. No wildcards are supported. (ex. *.example.com is not supported. If application uses such domain, it will have no effect)	O

With this API, app needs to register for SDSTATE_CHANGE_NOTIF notification to receive sd state.

II. API for Stopping sponsored data

This API is used for stopping sponsored data for the application. When the sponsored data is stopped, the SDK will notify the sdState change to application through SDSTATE_CHANGE_NOTIF notification.

The method signature is as follows:

+(void) stopSponsoredData;

III. API for re-Starting sponsored data

This API is used for restarting sponsored data after it is stopped. Calling this API without calling **initSponsoredData** API will result in no operation. When the sponsored data is re-started, the SDK will notify the sdState change to application through SDSTATE_CHANGE_NOTIF notification..

The method signature is as follows:

+(void) startSponsoredData;

IV. API to register custom NSURLSessionConfiguration

This API is needed with initSponsoredData for registering any custom session configuration used in the application.

The method signature is as follows:

+(void) registerAppConfiguration:(NSURLSessionConfiguration*) aConfig;

V. API for Single Content URL

This API is used for sponsoring individual url content on cellular networks. The method signature is as follows:

+(SmiResult*)getSDAuth:(NSString*)sdkKey url:(NSString*)url userId:(NSString*)userId;

getSDAuth API parameter details:

Name	Type	Description	[M]andatory/[O]ptional
sdkKey	String	The API Access Key provided by Datami	M
url	String	The specific url that needs to be sponsored	M
userId	String	The userId of the application. When provided, it can be used for analytics and/or authorization purposes.	O

V. SDK Analytics API

The Data SDK collects the following metrics:

Metric	Description
cellularSessionTime	Total application time in cellular mode (regardless of application is sponsored or not).
wifiSessionTime	Total application time in wifi mode.
sdDataUsage	Total sponsored data used by the application. This doesn't include the data usage in wifi.

This API is provided for application to fetch the above metrics on demand basis. The API signature is as follows:

+(SmiAnalytics*) getAnalytics;

1 Getting started with 'initSponsoredData' API

1.1 Call 'initSponsoredData' API in AppDelegate's 'didFinishLaunchingWithOptions' method

Objective C sample code for using getAppSDAuth API

```
#import "SmiSdk.h"
.......
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
 // Call the Datami API at the beginning of
didFinishLaunchingWithOptions, before other initializations.
    // IMPORTANT: If Datami API is not the first API called in the
application then any network
    // connection made before Datami SDK initialization will be
non-sponsored and will be
    // charged to the user.
 [SmiSdk initSponsoredData: mySdkKey userId: myUserId,
showSDMessage:YES];
 ....
}
```

Objective C sample code for using getAppSDAuth API with 'exclusionDomains' parameter

```
#import "SmiSdk.h"
.......
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
 // Call the Datami API at the beginning of
didFinishLaunchingWithOptions, before other initializations.
    // IMPORTANT: If Datami API is not the first API called in the
application then any network
    // connection made before Datami SDK initialization will be
non-sponsored and will be
    // charged to the user.
    NSArray *exclDomains = [NSArray arrayWithObjects:
@"www.example.com", @"www.google.com", @"images.google.com", nil];
 [SmiSdk initSponsoredData: mySdkKey userId: myUserId, showSDMessage:YES
exclusionDomains: exclDomains];
 ....
}
```

Swift sample code for using getAppSDAuth API

```
//in Swift
func application(application: UIApplication,
didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) ->
Bool
{
 // Call the Datami API at the beginning of
didFinishLaunchingWithOptions, before other initializations.
    // IMPORTANT: If Datami API is not the first API called in the
application then any network
    // connection made before Datami SDK initialization will be
non-sponsored and will be
    // charged to the user.
 SmiSdk.initSponsoredData(mySdkKey, userId:myUserId, showSDMessage:true)
}
```

Swift sample code for using getAppSDAuth API with 'exclusionDomains' parameter

```
//in Swift
func application(application: UIApplication,
didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) ->
Bool
{
 // Call the Datami API at the beginning of
didFinishLaunchingWithOptions, before other initializations.
    // IMPORTANT: If Datami API is not the first API called in the
application then any network
    // connection made before Datami SDK initialization will be
non-sponsored and will be
    // charged to the user.
    let exclDomains: NSArray = ["www.example.com", "www.google.com",
"images.google.com"]
 SmiSdk.initSponsoredData(mySdkKey, userId:myUserId, showSDMessage:true,
exclusionDomains:exclDomains as! [NSArray])

}
```

1.2 Call 'registerAppConfiguration' API for custom session configuration

This API is used to add custom URLSession configuaration. This needs to be called before creating any NSURLSession in the application.

sample code for regiserAppConfiguration API

```
#import "SmiSdk.h"


//........
 NSURLSessionConfiguration *config = [NSURLSessionConfiguration
defaultSessionConfiguration];
 [SmiSdk registerAppConfiguration:config];
```

1.3 Register a notification observer for Sponsored Data state changes in your AppDelegate implementation

Datami iOS SDK provides sponsored data state changes though 'SDSTATE_CHANGE_NOTIF' notification observer while application is running. The application can use current sponsored data state to display appropriate message to its users.

sample code to register notification for SD STATE

```
#import "SmiSdk.h"
.......


- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    [[NSNotificationCenter defaultCenter] addObserver:self

selector:@selector(receivedStateChage:)

name:SDSTATE_CHANGE_NOTIF object:nil];
 // Call the Datami API at the beginning of
didFinishLaunchingWithOptions, before other initializations.
    // IMPORTANT: If Datami API is not the first API called in the
application then any network
    // connection made before Datami SDK initialization will be
non-sponsored and will be
    // charged to the user.
 [SmiSdk initSponsoredData: mySdkKey userId: myUserId,
showSDMessage:YES];
 ....
}

...........
-(void)receivedStateChage:(NSNotification*)notif {
    SmiResult* sr =  notif.object;
    NSLog(@"receivedStateChage, sdState: %ld", (long)sr.sdState);
    if(sr.sdState == SD_AVAILABLE) {
        // TODO: show a banner or message to user, indicating that the
data usage is sponsored and charges do not apply to user data plan
    } else if(sr.sdState == SD_NOT_AVAILABLE) {
        // TODO: show a banner or message to user, indicating that the
data usage is NOT sponsored and charges apply to user data plan
        NSLog(@"receivedStateChage, sdReason %ld", (long)sr.sdReason);
    } else if(sr.sdState == SD_WIFI) {
        // wifi connection
    }
}
```

Swift

```
func application(application: UIApplication,
didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) ->
Bool {

 // Call the Datami API at the beginning of
didFinishLaunchingWithOptions, before other initializations.
    // IMPORTANT: If Datami API is not the first API called in the
application then any network
    // connection made before Datami SDK initialization will be
non-sponsored and will be
    // charged to the user.

    NSNotificationCenter.defaultCenter().addObserver(
   self,
   selector: #selector(receivedStateChage),
            name: SDSTATE_CHANGE_NOTIF,
            object: nil)

 SmiSdk.initSponsoredData(mySdkKey, userId:myUserId, showSDMessage:true)

    return true

 }
@objc func receivedStateChage(notification: NSNotification){
    let sr = notification.object as! SmiResult
    if sr.sdState==SdState.SD_AVAILABLE {
      // TODO: show a banner or message to user, indicating that the
data usage is sponsored and charges do not apply to user data plan
    }
 else if sr.sdState==SdState.SD_NOT_AVAILABLE {
   // TODO: show a banner or message to user, indicating that the data
usage is NOT sponsored and charges apply to user data plan
    }
    else if sr.sdState==SdState.SD_WIFI {
  // wifi connection
    }

    }
```

1.4 Setup User Communication

There are two ways in which SDK shows user messaging for Sponsored Data. The Application can choose one, both or none of these options.

1.4.1 User Messaging using Toast

The sdk driven messaging is available when application passes 'showMessaging' = true in the 'initSponsoredData' API. In such case, SDK will

show a popup message if the application is in foreground and sponsored data is available (SdState = SD_AVAILABLE) for the application.

1.4.2 User Messaging using Notification

The SDK can show a notification in the notification drawer when the application is in foreground and sponsored data is available (SdState = SD_AVAILABLE) for the application.

Register for local notification in AppDelegate's 'didFinishLaunchingWithOptions' method:

sample code for user messaging

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
 ....
 UIUserNotificationType types = UIUserNotificationTypeBadge |
     UIUserNotificationTypeSound | UIUserNotificationTypeAlert;

 UIUserNotificationSettings *notifSettings =
     [UIUserNotificationSettings settingsForTypes:types categories:nil];
     [[UIApplication sharedApplication]
registerUserNotificationSettings:notifSettings];
 ...
}
```

Swift

```
let settings = UIUserNotificationSettings(forTypes: [.Alert, .Badge,
.Sound], categories: nil)
UIApplication.sharedApplication().registerUserNotificationSettings(setti
ngs)
```

1.5 Videos in UIWebView

Besides **initSponsoredData** API, the application needs to use SmiUIWebView for covering videos in a WebView. The SmiUIWebView is a sub-class of iOS UIWebView, providing user messaging for the videos when the video are not sponsored.

1.5.1 Using SmiUIWebView directly in your application

Follow the below sample code if your application uses the iOS UIWebView directly.

sample code for SmiUIWebview

```
//MyViewController.h
@interface MyViewController : UIViewController <UIWebViewDelegate>
@end

//MyViewController.m
@interface MyViewController()
@end
@implementation MyViewController
....
#pragma mark - View Methods
- (void)viewDidLoad
{
 [super viewDidLoad];
 SmiUIWebView *webView = [[SmiUIWebView alloc] initWithFrame:CGRectMake(0, 0,
myWidth, myHieght)];
 webView.delegate = self;
 NSURLRequest *requestObj = [NSURLRequest requestWithURL:myUrl];
 [webView loadRequest:requestObj];
}
....
//-------------------------------------------------------------------------
----------
// Web View Delegate Methods
// Note: no changes are required in your Web View Delegate Methods for covering
videos
//-------------------------------------------------------------------------
---------
#pragma mark - Web View Delegate Methods
....
```

1.5.2 Using sub-class of SmiUIWebView in your application

Follow the below sample code if your application uses a sub-class of iOS UIWebView.

sample code for SmiUIWebview

```
// MyUIWebView.h
#import <UIKit/UIKit.h>
@interface MyUIWebView : SmiUIWebView
...
@end
// MyUIWebView.m
#import "MyUIWebView.h"
@implementation MyUIWebView
....
- (id)initWithFrame:(CGRect)frame{
 self = [super initWithFrame:frame];
 if (self) {
 ...
 }
 return self;
}
- (void) setDelegate:(id<UIWebViewDelegate>)delegate{
 [super setDelegate:delegate];
 ....
}
```

2 Using the 'stopSponsoredData' API

Apps can use this API to stop sponsored data at any time during the application runtime. For example, this API will be useful if the app wants to provide sponsored data for only logged in users. The App can call this API when user logs out.

2.1 Sample code for using 'stopSponsoredData' API

Using 'stopSponsoredData' API

```
@implementation viewController


-(void)didLogout{
 NSLog(@"user is logged out, stopping sponsored data");
    [SmiSdk stopSponsorData];

}


@end
```

3 Using the 'startSponsoredData' API

Apps can use this API to start sponsored data at any time during the application runtime. For example, this API will be useful if the app wants to provide sponsored data for only logged in users. The App can call this API when user logs in.

3.1 Sample Code for using the 'startSponsoredData' API

Using 'startSponsoredData' API

```
@implementation viewController

-(void)didLogin{
 NSLog(@"user is logged in, starting sponsored data");
    [SmiSdk startSponsoredData];

}

@end
```

4 Getting started with 'getSDAuth' API

4.1 Call 'getSDAuth' API for intended URL

sample code for getSDAuth API

```
#import "SmiSdk.h"
 .......

 SmiResult* sr = [SmiSdk getSDAuth:mySdkKey url:myUrl userId:myUserId];
 // get sponsored data url
 NSString* sdUrl =  sr.url;

 if(sr.sdState == SD_WIFI) {
  // device is in wifi, sdUrl is same as original url
     NSLog(@"Device is in wifi");
 } else if (sr.sdState == SD_AVAILABLE) {
     NSLog(@"SD_AVAILABLE");
  // sdUrl is a sponsored data url
 } else if(sr.sdState == SD_NOT_AVAILABLE) {
     NSLog(@"SD_NOT_AVAILABLE reason:%d", sr.sdReason);
  // sdUrl is not sponsored, same as original url
 }
```

5 Using 'getAnalytics' API

Apps can use this api to fetch the SDK collected analytics at anytime after the app is notified with an sdState.

5.1 Sample Code using the 'getAnalytics' API

Using 'getAnalytics' API

```
//.....
SmiAnalytics *smi = [SmiSdk getAnalytics];


NSLog(@"wifiTime: %f, smi.fgWifiSessionTime);
NSLog(@"nCellularSessionTime: %f, smi.fgCellularSessionTime);
NSLog(@"sdData usage: %lld, smi.sdDataUsage);
```

6 Hybrid app integration

In case when the developers use Cordova to develop hybrid application for both Android and iOS, there needs to be special integration done at the native code for i.e. Java for Android and ObjectiveC for iOS. The integration for Hybrid iOS is explained.

6.1 Integration Steps

1. Include the Datami iOS SDK library (libsmisdk.a) and SmiSdk.h in your project.
2. Build the ios project with **cordova build ios.**

6.2 Getting started with initSponsoredData() API

1. Integrate initSponsoredData() API. (Follow Section 1)

6.3 Getting started with getSDAuth() API

When the app is using a plugin to play the video from the URL, the video will not be sponsored. To make it sponsored, the application needs to use the **getSDAuth()** in the plugin code. As the plugin code is open source, we can find the location where the plugin actually passes the URL to the media player. Here we are taking the example of StreamingMedia plugin for playing videos in hybrid app. Call the getSDAuth() api in correct location. Here we need to change the file **StreamingMedia.m** and the function **startPlayer().**

StreamingMedia.m

```
#import "SmiSdk.h"
...

@implementation StreamingMedia {
...
}
-(void)startPlayer:(NSString*)uri {
    SmiResult* sr = [SmiSdk getSDAuth:mySdkKey url:uri userId:myuserid];
    NSURL* url;
    if(sr.url){
        NSLog(@"Returned URL %@",sr.url);
        url = [NSURL URLWithString:sr.url];
    }else{
        url = [NSURL URLWithString:uri];
    }
 moviePlayer =  [[MPMoviePlayerController alloc]
initWithContentURL:url];
}
```

6.4 Sponsoring Video Inside WebView

If the hybrid app is loading Embedded youtube and facebook videos inside the app, the app need to use the SmiUIWebView class to make the videos sponsored or blocked depending on the configuration at the backend. As the Cordova Library loads the Custom WebView we need to extend the SmiUIWebView in **CDVViewController.m** at the location **{Project_name}/platforms/ios/CordovaLib/classes/**

CDVViewController.m

```
#import "SmiUIWebView.h"


@interface CDVViewController () {
    NSInteger _userAgentLockToken;
    CDVWebViewDelegate* _webViewDelegate;
}

- (UIWebView*)newCordovaViewWithFrame:(CGRect)bounds{
    return [[SmiUIWebView alloc] initWithFrame:bounds];
}
```

7. Changes needed for NSAppTransportSecurity in plist (for iOS 9.0 and above).

Datami sdk uses an http request for sponsored data eligibility check, which is performed by wireless carrier. The application needs to add datami domains in plist.

```
<key>NSAppTransportSecurity</key>
<dict>
 <key>NSExceptionDomains</key>
 <dict>
 <key>cloudmi.datami.com</key>
 <dict>
 <key>NSIncludesSubdomains</key>
 <true/>
 <key>NSTemporaryExceptionAllowsInsecureHTTPLoads</key>
 <true/>
 <key>NSTemporaryExceptionMinimumTLSVersion</key>
 <string>TLSv1.1</string>
 </dict>
 </dict>
</dict>
```

8 Testing and Verification

8.1 Initial Integration Testing

When SDK integration is complete, the following basic tests should be done by an application developer (before an active sponsored data campaign is configured in Datami Portal):

1. Open the application in WiFi and verify that the SDK notifies the application with sdState = SD_WIFI through SDSTATE_CHANGE_NOTIF notification.
2. Open the application in cellular mode and verify that the SDK notifies the application with sdState = SD_NOT_AVAILABLE and sdReason = SD_NOT_AVAILABLE_PROMOTION_NOT_FOUND through SDSTATE_CHANGE_NOTIF notification. The sdReason can vary, depending on current stage of the application provisioning in Datami backend. For example, if the application is not whitelisted, the sdReason would be different.
3. Switch between WiFi and cellular, and verify that the SDK notifies the sdState accordingly.

8.2 Sponsored Data Testing

The following tests can be carried out when SDK integration is complete and there is an active campaign configured in the Datami Portal:

1. Open the application in cellular mode and verify that the SDK notifies the application with sdState = SD_AVAILABLE through SDSTATE_CHANGE_NOTIF notification.
2. Use the application in cellular and verify that the data usage in Datami portal increases for the campaign.
3. Use the application in WIFI and verify the data usage in Datami portal doen't increase for the campaign.

Note: The integrated application needs to be shared with Datami and cellular operator for complete testsing and verification, including zero rating verification.

9 Performance

9.1 Logging Overhead

Datami SDK uses very minimal logging in its release version. The logging rate is less than 1 KB per hr.

9.2 Initialization Overhead

1. initSponsoredData API is asynchronous and it doesn't block the calling application.
2. getSDAuth API is synchronous as it needs to return a sponsored url. However there is no network call involved in this api when used after initSponsoredData API.

9.3 Runtime Overhead

The Datami SDK routes the traffic through Datami Gateway which adds a minor latency to the data flow of no more than 100ms. The Datami gateway(s) are deployed either within or close to wireless carrier core network to minimize the delays. However, Datami offers different deployment models to minimise network latencies. The application developer can request for testing the correct integration from Datami.

10 Best Practices

In order to achieve the best results from Datami SDK assuring best experience for the user :

1. The App developer should call the Datami SDK before making any network connections.
2. The app developer should let the SDK handle messaging the user about the sponsorship availability.

11 FAQ

When do I need to use initSponsoredData API?

You should use initSponsoredData API when you want entire application traffic (http/https) needs to be sponsored

Where should I call the initSponsoredData API in my application code?

The initSponsoredData API should be called before making any network call from the application. The best place for this API call is at the beginng of application's didFinishLaunchingWithOptions method, see code sample in section 1.1

When do I need to use getSDAuth API?

You should use getSDAuth API for sponsoring individual content like specific url. You also need to use this API along with initSponsoredData for sponsoring videos that played through native media player.

Where should I call the getSDAuth API?

First of all, the getSDAuth API is only used for sponsoring specific content, unless it is a media url played through native media player. This API should be called just before using the specific url in your application code.

How do I check whether SDK integration correct?

If your application is receiving SDSTATE_CHANGE_NOTIF notification then you can assume that the integration correct.

How does the SDK sponsor the application traffic?

The Datami SDK routes the traffic through Datami getway, if the subscriber and application are authorized for sponsored data. The data going through Datami gateway is not charged to subscriber's data plan as Datami gateways are whitelisted by cellular provider.

Should I use SDK user messaging or my own messaging?

Notifying the user about sponsored data is critical. The SDK provides basic user messaging through a Toast message as it doesn't include any UI. The application needs to implement an UI based user messaging for better visibility and user experience.

My application is global, but Datami is not enabled for all locations or cellular providers. Do I need to check the location and/or wireless operator before calling Datami API?

The Datami SDK API calls are no-op where Datami service is not enabled. The SDK checks with Datami backend and goes dormant for a period if the service not enabled for the given location or cellular provider. The dormant period is configurable from Datami backend and its default value is 48 hrs.

I have already implemented the API but do not see any user messaging?

Check if there is a campaign running with Datami for the country where you are trying to run the tests. If not please send a query to admin@datami.com

My app has videos inside the app, how do I make them sponsored?

In order to make any video Sponsored, please use URL API provided in the section 2.

I have used URL api for my videos, but my videos are still not sponsored, what am I missing?

This can happen if the application is using a third party library to play videos, please send the name of the library with your query to admin@datami.com

My application uses WebView to show content, how do I make sure the videos in the html are not getting sponsored?

Datami SDK provides a provision of blocking the videos on Webview, you can follow the section 1.5.

Can the SDK cause an application crash?

We have taken utmost care to ensure that the SDK will not cause an application crash. We have verified it by running the SDK on all possible devices and versions of the operating systems. After integration with our SDK, please test your application thoroughly by testing it on all versions of devices and operating systems before you release it to the App Store. You can use third-party services like Keynote, AWS, etc.

I use Hybrid application to develop iOS and Android app, how do I integrate the SDK?

If you are using Cordova as the platform to develop Hybrid Application you can follow section 3 to integrate the SDK.

In my Hybrid application I do not use StreamingMedia plugin to show video, where do I integrate the URL api in the code?

In most of the video plugins, they use simple **[[MPMoviePlayerController alloc] initWithContentURL:url];** function to show video on the device. If you can find the location of that, you can integrate the API at that location. If not, then you can mail the name of the plugin to admin@datami.com and we will get back to you with the solution.

I am Hybrid application developer, and not aware of the Objective C code for the iOS development, how do I integrate the SDK?

We have made the SDK integration document easy to understand and integrate. But if you face any issues, you can mail your query at admin@datami.com and we will get back to you.

P.S. we will soon be coming up with Cordova plugin to make the integration easier.