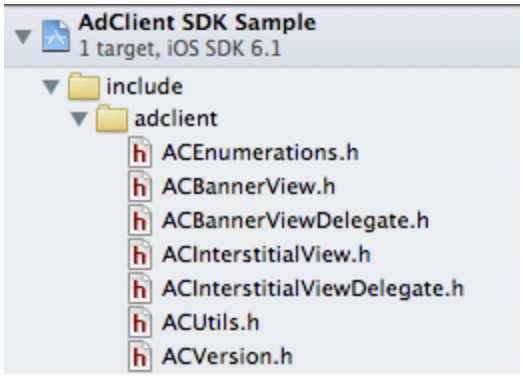
AdClient iOS SDK

User Manual

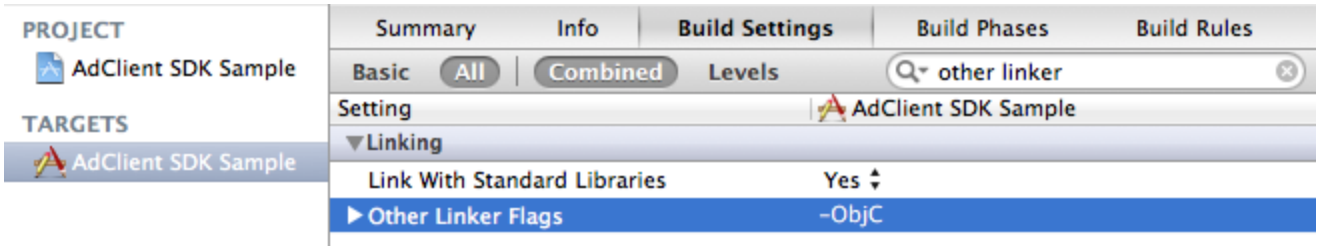
This manual provides a simple step by step guidance for a quick and painless integration of AdClient advertising into your applications.

1. Setup

Unpack and place AdClient SDK files into the root directory of your project. Copy files from AdClient SDK/include/ and the library file AdClientSDK/lib/libadclient.a to the project, as shown in a picture below:



After that, navigate to 'targets' select the 'build settings' tab and add '-ObjC:



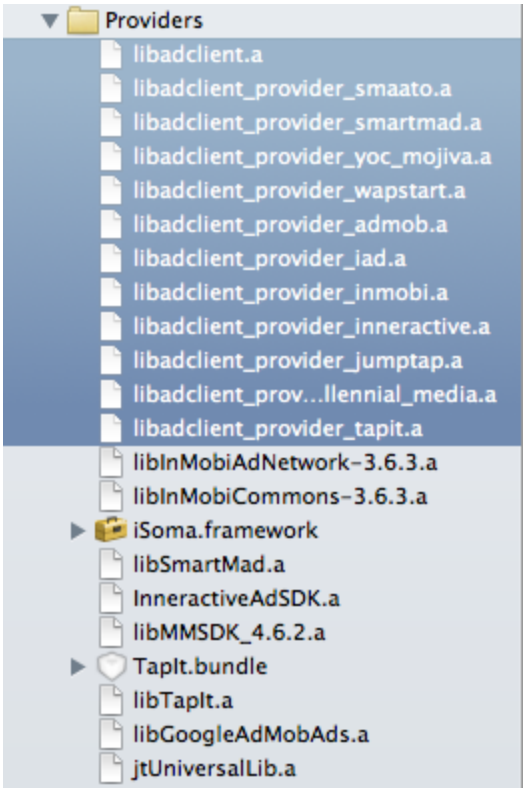
Minimum supported version of iOS for AdClient SDK is 4.3.

2. Specifying AdClient SDK Ad Providers

Next, specify which ad network providers will be used. Each ad network provider is implemented as a static library, so you are able to link with the particular providers you chose instead of all of them. **Note that the providers library does not include ad network SDK, but only contains an interface for interaction between the AdClient SDK core and ad network SDK.**

For example, if you need to attach Google AdMob banners, you have to add a static library AdClientSDK/lib/libadclient_provider_admob.a and Google AdMob libGoogleAdMobAds.a file. AdClient SDK cannot provide external dependencies due to legal issues.

After adding AdClient providers to your project, the project tree will look somewhat like this:



Currently, the following ad networks are supported:

- Google AdMob - version 6.12.0 checked
- Apple iAd (note that you will need to add iAd.framework to your project) - iOS 7.1 checked
- MillennialMedia - version 5.1.1 checked
- InMobi - version 4.1.0 checked
- Inneractive - version 2.0.7 is checked
- Jumptap - version 2.0.16.0 checked
- YocPerformance - version 3.1.2 built in
- TapIt - version 3.0.1 checked

- WapStart - version 2.1.3 checked (not integrated, comes as sources)
- iSoma (Smaato) - version 6.1.0 checked
- Mojiva - version 3.1.2 built in
- Amazon Ads

Also, ad network SDKs commonly depend on iOS native frameworks. Linking your app against frameworks does not increase your application's executable size, so you can just put them all into your application project. Here is the full list of iOS frameworks that are needed for the current list of ads providers:

- AdSupport.framework
- AudioToolbox.framework
- AVFoundation.framework
- CFNetwork.framework
- CoreFoundation.framework
- CoreGraphics.framework
- CoreLocation.framework
- CoreMotion.framework
- CoreTelephony.framework
- EventKit.framework
- EventKitUI.framework
- iAd.framework
- ImageIO.framework
- MediaPlayer.framework
- MessageUI.framework
- MobileCoreServices.framework
- QuartzCore.framework
- PassKit.framework
- Security.framework
- StoreKit.framework
- SystemConfiguration.framework
- UIKit.framework
- libsqlite3.dylib
- libz.dylib
- libxml2.dylib

3. Network Account Options

AdClient iOS SDK needs an ads-server url to be set up. To do this, use the setAdsServerUrl:(NSString *) of **ACUtils** class (ACUtils.h header file). Note, that the trailing slash in your new url is important.

4. Basic Banner Integration

For basic integration you will need to create and attach ACBannerView-class

object into your main view. You can create as many objects of class ACBannerView as you need. Here is a small tutorial that will help you do this:

a. First of all, import ACView.h header into you view controller class code

```
#import "ACBannerView.h"
```

b. Create an object of class ACBannerView with the following parameters

```
ACBannerView *acView = [[[ACBannerView alloc] initWithID:@"YOUR_PLACEMENT_ID"
                                                sizeType:ACBannerViewSize320x50
                                      modalViewController:self
                                              useLocation:YES
                                                 testMode:YES] autorelease];
```

where

ID - key, provided at registration with AdClient;

sizeType - size of banners. Currently the supported sizes are: 320x50 for iPhone applications and 728x90 for iPad applications;

modalViewController - view controller for modal view ad presenters. Can not be nil;

useLocation - force turn on location check for ads or use passive mode (only if app is using location tracker);

testMode - do or do not use test mode (note that some ad networks can not send ads for your test running app in iOS Simulator without this mode switched 'On').

c. Attach the recently created ACBannerView object to main view:

```
[mainView addSubview:acBannerView];
```

That's all. Now your application can run ads provided by AdClient.

5. Advanced Banner Integration

a. If you need to change the ads' refresh time (which is set to 15 seconds by default), you can access the ACBannerView property refreshTimeInterval. Minimum refresh time is 5 seconds.

```
acBannerView.refreshTimeInterval = 10.0;
```

b. If you need to change AdClient SDK for iOS at the log level, or silence it, you can use ACUtils class.

```
#import "ACUtils.h"
…
[ACUtils setLogLevel:ACVerboseNone];
…
```

c. Possibly, your app will need the ads' click-time lifecycle notifications. For getting that you need to derive all your classes from ACBannerViewDelegate (AdClient SDK/include/adclient/ACBannerViewDelegate.h) and implement the following notifications callbacks:

```
// Notification: Banner ad started loading.
-(void)acBannerViewDidStartLoadAd:(ACBannerView *)acBannerView;

// Notification: Banner ad failed to load.
// Error code/type is output to debug console
-(void)acBannerViewDidFailLoadAd:(ACBannerView *)acBannerView;

// Notification: Banner ad successfully loaded.
-(void)acBannerViewDidLoadAd:(ACBannerView *)acBannerView;

// Notification: Ad provided by ACBannerView will be shown
-(void)acBannerViewWillShowAd:(ACBannerView *)acBannerView;

// Notification: Ad provided by ACBannerView is shown
-(void)acBannerViewDidShowAd:(ACBannerView *)acBannerView;

// Notification: Ad provided by ACBannerView has been tapped
-(void)acBannerViewAdHasBeenClicked:(ACBannerView *)acBannerView;

// Notification: Ad provided by ACBannerView will enter modal mode when opening embedded  screen view
controller
-(void)acBannerViewWillEnterModalMode:(ACBannerView *)acBannerView;

// Notification: Ad provided by ACBannerView did leave modal mode
-(void)acBannerViewDidLeaveModalMode:(ACBannerView *)acBannerView;

// Notification: Ad provided by ACBannerView causes to leave application to navigate to Safari, iTunes, etc.
-(void)acBannerViewWillLeaveApplication:(ACBannerView *)acBannerView;
```

d. If you need to customize ad requests by additional parameters (age, income, gender), you can use the customParameters property of ACBannerView class objects (see ads customization section in AdClient account setup documentation). For example:

```
acBannerView.customParameters = [NSDictionary dictionaryWithObject:@”dog” forKey:@”pet”];
```

e. If you would like banner views to open landing pages in an external browser (not through the embedded in-app web view controller), you can use the preferExternalBrowser boolean property of ACBannerView class objects. PLEASE NOTE, that only a few of the supported advertisers’ sdks allow to modify their landing page behavior!

6. Interstitials Integration

AdClient SDK for iOS provides another method to display ads - interstitials. Interstitial ad view is fullscreen, and the SDK user controls the ad’s lifetime and when interstitial have to be shown. For receiving:

```
#import “ACInterstitialView.h”
…
ACInterstitialView *acInterstitialView = [[ACInterstitialView alloc]
                                                    initWithID:@"53927211d9604e5d671963fd0
                                                13dd94b"
                                                    useLocation:YES
```

```
                                                    testMode:YES];
...
```

The code above creates an interstitial ad, which automatically starts to load the ad, but does not display it. Interstitial ads can only be displayed if they already loaded. For this, your container class must conform to the ACInterstitialViewDelegate protocol:

```
 // Notification: Interstitial ad started loading. State is changed to ACInterstitialViewStateLoading.
-(void)acInterstitialViewDidStartLoadAd:(ACInterstitialView *)acInterstitialView;

// Notification: Interstitial ad failed to load. State is changed to ACInterstitialViewStateFailed.
// Error code/type is output to debug console
-(void)acInterstitialViewDidFailLoadAd:(ACInterstitialView *)acInterstitialView;

// Notification: Interstitial ad successfully loaded. State is changed to ACInterstitialViewStateReady.
-(void)acInterstitialViewDidLoadAd:(ACInterstitialView *)acInterstitialView;

// Notification: Interstitial ad starts to display ad. State is changed to ACInterstitialViewStateActive.
-(void)acInterstitialViewWillEnterModalMode:(ACInterstitialView *)acInterstitialView;

// Notification: Interstitial ad finished to display ad. State is changed to ACInterstitialViewStateDone.
-(void)acInterstitialViewDidLeaveModalMode:(ACInterstitialView *)acInterstitialView;

// Notification: User has interacted with ad provided by ACInterstitialView. Optional is application leaving to
navigate to Safari, iTunes, etc.
-(void)acInterstitialViewUserInteraction:(ACInterstitialView *)acInterstitialView
willLeaveApplication:(BOOL)yesOrNo;
```

assign itself as delegate for ACInterstitialView via `acInterstitialView.delegate = self;` and use messages `acInterstitialViewDidLoadAd:` and `acInterstitialViewDidFailLoadAd::`

```
-(void)acInterstitialViewDidFailLoadAd:(ACInterstitialView *)acInterstitial
{
        // just release object
        [acInterstitialView release];
}

-(void)acInterstitialViewDidLoadAd:(ACInterstitialView *)acInterstitial
{
        // present acInterstitial with current view controller
        [acInterstitial presentWithViewController:self];
}

-(void)acInterstitialViewDidLeaveModalMode:(ACInterstitialView *)acInterstitial
{
        // Interstitial is done. Release it

        [acInterstitial release];
}
```

7. Application Startup Interstitial

Another option for using interstitials is the application startup splash ad. To use it, add the following code in your application delegate method application:didFinishLaunchingWithOptions: after the initialization and keymaking of main window:

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions
{
    …
        [self.window makeKeyAndVisible];

        // set up interstitial view
        ACInterstitialView *acInterstitialView = [[ACInterstitialView alloc]
                                                  initWithID:@"53927211d9604e5d671963fd0
                                                  13dd94b"
                                              useLocation:NO
                                                testMode:YES];
        acInterstitialView.loadTimeout = 4.0;
        acInterstitialView.delegate = self;
        [acInterstitialView presentAsStartupScreenWithWindow:self.window
                defaultImage:[UIImage imageNamed:@"InterstitialStartup.png"]];
        return YES;
}
```

and control its lifetime via notifications

```
// Notification: Interstitial ad failed to load. State is changed to ACInterstitialViewStateFailed.
// Error code/type is output to debug console
-(void)acInterstitialViewDidFailLoadAd:(ACInterstitialView *)acInterstitialView
{
        [acInterstitialView release];
}


// Notification: Interstitial ad finished to display ad. State is changed to ACInterstitialViewStateDone.
-(void)acInterstitialViewDidLeaveModalMode:(ACInterstitialView *)acInterstitialView
{
        [acInterstitialView release];
}
```