



**Getting
Started Guide
– YuMe
iPhone®-iPad®
SDK (3.2.x.6)**

Table of Contents

Introduction	2
YuMe Implementation Process	2
Integration Instructions.....	4
iOS Version Support.....	4
SDK setup	4
iPhone-iPad SDK includes	4
Preparing your iPhone or iPad application to use SDK.....	4
Plug-n-Play Integration - YuMe Recommended.....	8
1. Initializing the SDK.....	8
Sample Code	9
2. Turn On Ad Pre-fetch and Caching – initAd()	9
Sample Code for initAd()	9
3. Play ads – showAd()	10
Sample Code for showAd()	10
4. Check for Ad play Complete – adComplete()	11
Sample Code for adComplete().....	11
5. Log Ad Events for debugging	11
6. Handle Device Orientation Changes	12
7. Release the SDK	13
Advanced Integration.....	14
Initializing the SDK.....	14
SDK management APIs	15
Flow Control APIs	18
Other APIs	18
Log Ad Events for debugging	19
Integration Support	21
Appendix.....	22
Plug-n-Play – API call sequence	22
Plug-n-Play Integration – Comprehensive Sample Code.....	22

Introduction

Welcome to YuMe, and thank you for deciding to become a YuMe mobile publisher/app developer. To start monetizing mobile inventory you'll need to integrate the iPhone-iPad SDK with your application. We want to make this process as fast and as easy as possible, so please email us at yume_support@yume.com if you run into issues.

YuMe Implementation Process

Step 1: Determine ad experience – Publishers' responsibility

Before you begin integration, you need to determine where you wish to insert a video ad into your application. YuMe recommends inserting a video ad shortly after the app launches, and possibly as an interstitial later on during app use. YuMe ads that run on iPhones are full-screen ads at 320x480. The minimum accepted size on iPad is 320x480.

Step 2: Complete the integration – Publishers' responsibility

Integrate the simulator assembly into your application. When this is complete, please send us the application and we will QA it to make sure ads are running properly. Once this is complete, you will swap out the simulator assembly for the device assembly, and then submit to Apple® for approval.

Integration is estimated to be a 2-3 day effort. The estimate does not account for app development.

Step 3: Integration QA – Joint effort, YuMe & Publisher

In order for us to certify your integration, we need to run a few tests on it. This includes making sure that each of our ad units display properly in your application. Please send the simulator application to yume_support@yume.com and be sure to note any app specific settings we'll need in order to validate it.

Step 4: Start monetizing your mobile content

As a YuMe Publisher/app developer, you will be able to receive video ads and other high-impact ad units from YuMe.

Once your integration has passed QA and we have reviewed your content and verified that it does not violate our standards, we will give you a login for our web-based campaign reporting console and start sending you ads. We will also work with you to determine which content and audience channels your videos or content should be assigned to, and determine how much inventory you will want YuMe to monetize on a daily basis. Once your content is assigned to YuMe channels, we will send you ads from the YuMe Network.

Integration Instructions

iOS Version Support

YuMe iOS SDK supports Apple iOS 4.0.2 and above

NOTE:

Before you start with integration, please note that after your fully developed app has been approved by YuMe QA for integration, you must still submit the app to Apple for approval. Apple still needs to approve your application for distribution via the App StoreSM. App submission and securing Apple's approval is the full responsibility of the publisher/app developer.

In addition, it is your responsibility to ensure that your application complies with all requirements set forth in your iOS SDK license agreement and any other agreements between you and Apple, including without limitation requirements relating to advertising.

SDK setup

iPhone-iPad SDK includes

- **libYuMeSDK_xxx.a** - Binary that serves ads. We supply a simulator build that can be used for testing in the simulator and a device build that can be used for testing on the device
- **YuMeSDK.h** - Header file that exposes the YuMe SDK interface.
- **YuMeAdParams.h** – Class used by applications to pass the necessary parameters (like ad server URL) to the SDK.
- **YuMeSDKDelegate.h** - Protocol that applications must implement in order to receive ad events from the SDK.
- **YuMeTypes.h** - Declares constants that are used by the SDK.
- **libical_xxx.a** – Binary file that supports calendar functionality. We supply a simulator build that can be used for testing in the simulator and a device build that can be used for testing on the device

Preparing your iPhone or iPad application to use SDK

1. Copy the following files to your project folder:

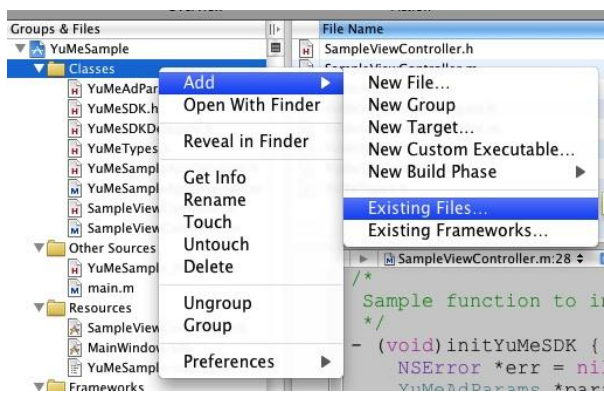
```
libYuMeSDK_Device.a  
libYuMeSDK_Simulator.a  
YuMeSDK.h  
YuMeAdParams.h
```

YuMeSDKDelegate.h
YuMeTypes.h
libical_Device.a
libical_Simulator.a

2. Add header files to Xcode project

- Go to project explorer
- Right click on the Classes folder and then Select Add→ Existing Files
- Select **YuMeSDK.h**, **YuMeAdParams.h**, **YuMeSDKDelegate.h**, **YuMeTypes.h**

Visual – Adding header files to project



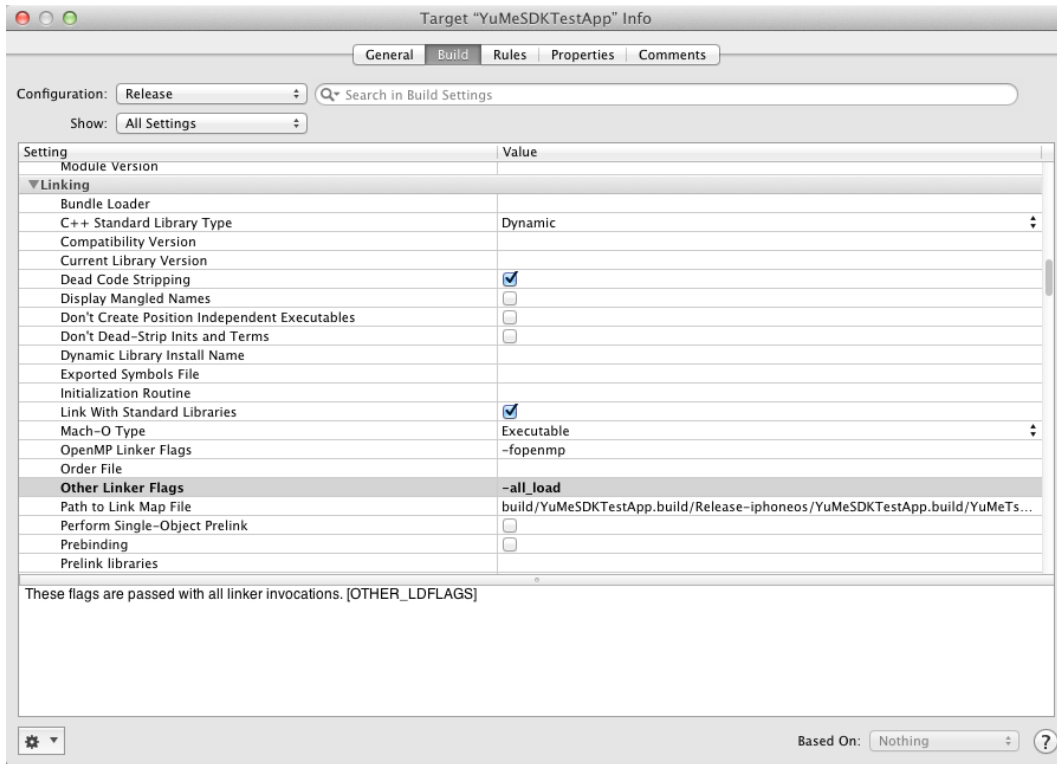
3. Add the following library files to Xcode project.

- Go to project explorer.
- Right click on the Frameworks folder and then select Add->Existing Files...
- Select **libYuMeSDK_Device.a**, **libYuMeSDK_Simulator.a**, **libical_Device.a**, **libical_Simulator.a**

4. Add the following to Xcode project

- Go to Project menu.
- Select Edit Active Target-> Build tab.
- Set **-all_load** in Other Linker Flags

Visual – Adding header files to project



5. Add the following to Xcode project

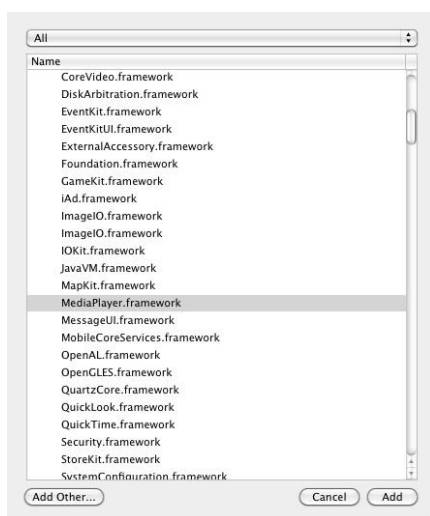
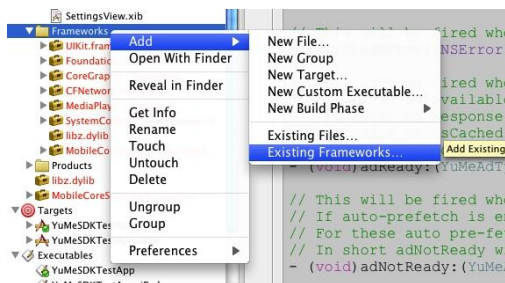
- Go to project explorer.
- Right click on the Frameworks folder and then select Add->Existing Frameworks...

Select **CFNetwork.framework**, **MediaPlayer.framework**, **CoreGraphics.framework**, **SystemConfiguration.framework**, **libz.dylib**, **MobileCoreServices.framework**, **CoreTelephony.framework**, **EventKit.framework (optional)**

Note: The EventKit framework is utilized for rich ad units to add events to Apple's native calendar application upon user interaction and consent. The SDK only writes events - and not read.

The Eventkit framework is detected by the YuMe SDK during application runtime. If the Eventkit framework is **NOT** included as a part of the build, the SDK will still allow the project to compile.

Visual – Adding frameworks to Xcode project



NOTE:

During the build you may see a few warnings that can be safely ignored

Example Warnings:

ld: warning: in .../libYuMeSDK_simulator.a, file was built for unsupported file format which is not the architecture being linked (armv7)

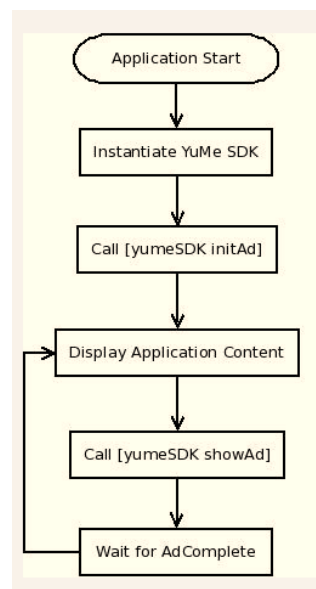
These warnings appear because we have added both libYuMeSDK_Device.a and libYuMeSDK_Simulator.a to the project. Only one of libYuMeSDK_xxx.a will be used at a time based upon the active build configuration.

6. Implement the protocol **YuMeSDKDelegate** and handle all the delegate methods in it. This is typically done in the same class where you initialize the SDK.

Plug-n-Play Integration - YuMe Recommended

Plug-n-Play Integration involves 7 steps

1. Initializing the SDK
2. Turn on Ad Pre-fetch and Caching – `initAd()`
3. Play ads – `showAd()`
4. Check for ad play complete – `adComplete()`
(Steps 1 through 4 depicted in the flow chart)
5. Log ad events for debugging
6. Handle Device Orientation Changes
7. Release the SDK at application exit



NOTE:

- If you are going to count ad opportunities, please only count an opportunity when you call `showAd()`.

1. Initializing the SDK

- Create an instance of **YuMeAdParams** and set the properties.
- Create an instance of **YuMeSDK** and initialize it using the **YuMeAdParams** object.

Default **YuMeAdParams** settings and values to set

- Domain ID = @"704oIaHzpGu";
- `yumeAdServerURL` = @"http://shadow01.yumenetworks.com/";
- `storageSize` = 5; // (5MB)

NOTE:

- The value for **params.domain** and **params.yumeAdServerURL** will change after QA has been completed. We will send you production values for these two parameters.
- Handle error cases.

You need to instantiate the SDK only once. It can be used to fetch ads until it's released

Sample Code

```
- (void)initYuMeSDK {
    NSError *err = nil;
    YuMeAdParams *params = [[YuMeAdParams alloc] init];

    params.domain = @"704oIaHzpGu";
    params.yumeAdServerURL = @"http://shadow01.yumenetworks.com/";
    params.storageSize = 5;

    yumeSDK = [[YuMeSDK alloc] initWithParams:params
errorStatus:&err];
    [params release];

    if (!yumeSDK) {
        NSString *errString = [YuMeSDK getErrDesc:err];
        NSLog(@"YuMe SDK initialization failed: %@", errString);
    }
    else {
        [yumeSDK setDelegate:self];
        NSLog(@"YuMe SDK was initialized successfully");
        // This function will initiate pre-fetching of Ads in the
background.
        [self initiateAdPreFetch];
    }
}
```

2. Turn On Ad Pre-fetch and Caching – initAd()

Call the **initAd** method with the type of ad you need at the time of app initiation. (these are defined in YuMeTypes.h). – Sample Code below

Note: **initAd** method should be called only once after step 1. initAd() should not be called more than once. SDK continues to pre-fetch ads following the completion of an ad play until the release of SDK after the first initAd() call. It is highly recommended to call this method immediately after app loading for optimal performance

Sample Code for initAd()

```
- (void)initiateAdPreFetch {
    NSError *err = nil;
    BOOL b = [yumeSDK initAd:YuMeAdTypePreRoll errorStatus:&err];
}
```

```

if (!b) {
    NSString *errString = [YuMeSDK getErrDesc:err];
    NSLog(@"initAd failed: %@", errString);
}
else {
    NSLog(@"initAd returned successfully");
}
}

```

3. Play ads – showAd()

NOTE: Please allow enough time after step 2 (initAd) and before calling step 3 (showAd) to ensure that the ad is pre-fetched and cached in memory to play. YuMe recommendation: 40 seconds.

Required Settings:

Guideline for parent set view using setParentView method

The view:

- should be equal to or less than the device screen size
- should be aligned with screen size to be fully visible on the device screen
- should not be used to display content – use separate views for content and ad

Call the **showAd** method with a pointer to an NSError object. This will begin playing the ad that was pre-fetched and cached. – Sample Code below

Note that showAd call needs to be called every time an ad needs to play.

Possible callbacks:

- adPlaying – ad is playing.
- adCompleted – ad has completed playing.
- adExpired – if ad is Expired
- adError followed by adCompleted – when an error occurs.

Sample Code for showAd()

```

- (void)displayAd {
    YuMeAdTypeEnum adType = YuMeAdTypePreRoll;

    NSError *err = nil;

    NSLog(@"Ad available for: %d", adType);
}

```

```

        // If needed, replace self.view with application created
view where the Ad needs to be displayed
        [yumeSDK setParentView:self.view];
        BOOL b = [yumeSDK showAd:adType errorStatus:&err];

        if (!b) {
            NSString *errString = [YuMeSDK getErrDesc:err];
            NSLog(@"showAd failed: %@", errString);
            // Application continues with content
        }
        else {
            NSLog(@"showAd returned successfully");
            // Application waits for adCompleted event
        }
    }
}

```

4. Check for Ad play Complete – adComplete()

Check for ad play complete. Application needs to move to content after an ad play is complete and hence the check.

Sample Code for adComplete()

```

- (void)adCompleted {
    NSLog(@"Ad completed");
    // Publisher takes control
}

```

5. Log Ad Events for debugging

Please implement below for logging purposes which will be required for debugging any errors

Delegate Methods

Implement the YuMeSDKDelegate methods in your implementation file.

Sample code

```

- (void)adPresent {
    NSLog(@"Ad present");
}

- (void)adAbsent {
    NSLog(@"Ad absent");
}

```

```

- (void)adPlaying {
    NSLog(@"Ad playing");
}

- (void)adExpired: (YuMeAdTypeEnum) adType {
    NSLog(@"Ad Expired");
}

- (void)adCompleted {
    NSLog(@"Ad completed");
    // Application continues with content
}

- (void)adError: (NSError *)error {
    NSLog(@"Ad error");
}

- (void)adReady: (YuMeAdTypeEnum) adType assetsCached: (BOOL) isCached {
    NSLog(@"Ad ready: %@", (isCached ? @"TRUE" : @"FALSE"));
}

- (void)adNotReady: (YuMeAdTypeEnum) adType {
    NSLog(@"Ad not ready");
}

```

6. Handle Device Orientation Changes

If the application supports Orientation changes, the parent view given to YuMe SDK should be re-sized by the application accordingly. Resizing is important to ensure the ad is displayed appropriately

Invoke the below call every time the device orientation changes. (Vertical to Landscape or vice versa).

Re-sizing of parent view can be done in the following functions:

```

(void)willAnimateRotationToInterfaceOrientation: (UIInterfaceOrientation)
toInterfaceOrientation duration: (NSTimeInterval)duration

(void)didRotateFromInterfaceOrientation: (UIInterfaceOrientation)fromIn
terfaceOrientation

```

If didRotateFromInterfaceOrientation is used, application has to call the following function in the SDK.

```
[yumeSDK orientationChanged];
```

If `willAnimateRotationToInterfaceOrientation` is used, the SDK will automatically detect the changes.

7. Release the SDK

Release the SDK before exiting the application

Call `[yumeSDK release]`

Advanced Integration

Initializing the SDK

- Create an instance of **YuMeAdParams** and set the properties.
 - Create an instance of **YuMeSDK** and initialize it using the **YuMeAdParams** object.
- Default **YuMeAdParams** settings and values to set

```
// To store the publisher domain.
o NSString *domain = @"704oIaHzpGu";

// To store the YuMe Ad server URL.
o NSString *yumeAdServerURL =
@"http://shadow01.yumenetworks.com/";

// The playlist response timeout value in seconds.
// Valid value is between 4 and 60 including.
// Default value is 5 (if timeOut < 4 default will be used).
// If timeOut is > 60 error will be returned.
o NSInteger timeout = 5;

// Time out value for interruption during ad streaming.
// Default value is 6 (if value < 3, default will be used).
// Valid value is between 3 and 60 including.
// If value is > 60 error will be returned.
o NSInteger streamingTimeOut = 6;

// Optional. Any additional parameters given here will be added to the query
string part of the Ad request. For example: gender=m
// Default is NULL @"
o NSString *additionalParams = @"";
o NSString *additionalParams = @"gender=m";

// Disk quota in MB. If this value is <= 0 caching cannot be turned on even if
setCacheEnabled is set to TRUE.
o float storageSize = 5;
```

NOTE:

- The value for **params.domain** and **params.yumeAdServerURL** will change after QA has been completed. We will send you production values for these two parameters.
- Handle error cases.

You need to instantiate the SDK only once. It can be used to fetch ads until it's released

Initialize the SDK - Sample Code

```
- (void)initYuMeSDK {
    NSError *err = nil;
    YuMeAdParams *params = [[YuMeAdParams alloc] init];

    // Populate Params
    params.domain = @"704oIaHzpGu";
    params.yumeAdServerURL = @"http://shadow01.yumenetworks.com/";
    params.additionalParams = @"";
    params.storageSize = 5;

    yumeSDK = [[YuMeSDK alloc] initWithParams:params
    errorStatus:&err];
    [params release];

    if (!yumeSDK) {
        NSString *errString = [YuMeSDK getErrDesc:err];
        NSLog(@"YuMe SDK initialization failed: %@", errString);
    }
    else {
        [yumeSDK setDelegate:self];
        NSLog(@"YuMe SDK was initialized successfully");
    }
}
```

SDK management APIs

➤ Initialize the SDK

```
NSError *err = nil;
YuMeAdParams *params = [[YuMeAdParams alloc] init];
// Populate params
YuMeSDK *yumeSDK = [[YuMeSDK alloc] initWithParams:params
errorStatus:&err];
```

➤ Release the SDK

```
[yumeSDK release]
```

➤ Initiate pre-fetch and caching of ads – Should call only once during an app session


```
[yumeSDK initAd:type errorStatus:&err]
```

Possible Events:

- adReady (and the isCached property is True) - ad available, fully downloaded and ready to play by calling the showAd method.
- adReady (and the isCached property is False) - ad available but not ready for play (in the middle of download)
- adNotReady- ad not available (similar to the adAbsent event)
- adError followed by adCompleted - when an error occurs.

adReady will be invoked twice, first with FALSE and then with TRUE. showAd() should be called only after adReady(TRUE).

adReady will be invoked only for first pre-fetch (or first time initAd is called) and not for subsequent pre-fetches. Hence adReady(TRUE) should not be used as a condition for calling showAd()

➤ **Play a pre-fetched and cached ad**

```
[yumeSDK showAd:&err];
```

Possible callbacks:

- adPlaying - ad is playing.
- adCompleted - ad has completed playing.
- adExpired - if ad is Expired
- adError followed by adCompleted - when an error occurs.

➤ **Stop a playing ad**

```
[yumeSDK stopAd:&err];
```

➤ **Disable auto pre-fetch**

```
[yumeSDK setAutoPrefetch:FALSE];
```

➤ **Disable caching of Ad creatives:**

```
[yumeSDK setCacheEnabled:FALSE];
```

- Clear Cached Ads and start with fresh auto ad pre-fetch

```
NSError *err = nil;
[yumeSDK clearCache:&err];
```

- Clear Cached Ads and stop auto pre-fetch

```
NSError *err = nil;
[yumeSDK setAutoPrefetch:FALSE]; // These 2 calls must be
called in the same order.
```

```
[yumeSDK clearCache:&err];
```

- Play ad via non pre-fetch method

```
[yumeSDK setParentView:adView]; // if needed, replace adView
with your parent view

NSError *err = nil;
BOOL b = [yumeSDK startAd:YuMeAdTypePreRoll errorStatus:&err];

if (!b) {
    NSString *errString = [YuMeSDK getErrDesc:err];
    NSLog(@"startAd failed: %@", errString);
}
else {
    NSLog(@"startAd returned successfully");
}
```

- Enable controlbar toggle for video & image ads

```
[yumeSDK setControlBarToggle:YES];
```

By default top and bottom control bar will hide after 5 seconds.

- Disable controlbar toggle for video & image ads

```
[yumeSDK setControlBarToggle:NO];
```

Top and bottom control will not hide.

Flow Control APIs

- **Pause Downloads.** Can be resumed later

```
[yumeSDK pauseDownload];
```

- **Resume Download**

```
[yumeSDK resumeDownload];
```

- **Abort Download**

```
[yumeSDK abortDownload];
```

- **Download status**

```
YuMeDownloadStatusEnum status = [yumeSDK getDownloadStatus];
```

- **Download Percentage**

```
float percentage = [sdk getDownloadedPercentage];
```

- **Clear Cookies**

```
[sdk clearCookies];
```

- **Get SDK Version**

```
NSString *version = [sdk getVersion];
```

Other APIs

- **Modify ad request parameters**

```
NSError *err = nil;  
YuMeAdParams *params = [yumeSDK getParams];  
// Modify the required parameters
```

```

BOOL b = [yumeSDK modifyParams:params errorStatus:&err];

if (!b) {
    NSString *errString = [YuMeSDK getErrDesc:err];
    NSLog(@"modifyParams failed: %@", errString);
}
else {
    NSLog(@"modifyParams returned successfully");
}

```

➤ Ad display as per device orientation

If the application supports Orientation changes, the parent view given to YuMe SDK should be re-sized by the application accordingly. Resizing is important to ensure the ad is displayed appropriately

Re-sizing of parent view can be done in the following functions:

```

(void)willAnimateRotationToInterfaceOrientation:(UIInterfaceOrientation)toInterfaceOrientation duration:(NSTimeInterval)duration

```

```

(void)didRotateFromInterfaceOrientation:(UIInterfaceOrientation)fromInterfaceOrientation

```

If didRotateFromInterfaceOrientation is used, application has to call the following function in the SDK.
[yumeSDK orientationChanged];

If willAnimateRotationToInterfaceOrientation is used, the SDK will automatically detect the changes.

Log Ad Events for debugging

Sample Code

```

#import "YuMeSDK.h"
#import "YuMeSDKDelegate.h"

@interface SampleViewController : UIViewController <YuMeSDKDelegate> {
    YuMeSDK *yumeSDK;
}

```

Delegate Methods

Implement the YuMeSDKDelegate methods in your implementation file.

Sample code

```
- (void)adPresent {
    NSLog(@"Ad present");
}

- (void)adAbsent {
    NSLog(@"Ad absent");
}

- (void)adPlaying {
    NSLog(@"Ad playing");
}

- (void)adCompleted {
    NSLog(@"Ad completed");
    // Application continues with content
}

- (void)adError:(NSError *)error {
    NSLog(@"Ad error");
}

- (void)adReady:(YuMeAdTypeEnum)adType assetsCached:(BOOL)isCached {
    NSLog(@"Ad ready: %@", (isCached ? @"TRUE" : @"FALSE"));
}

- (void)adNotReady:(YuMeAdTypeEnum)adType {
    NSLog(@"Ad not ready");
}

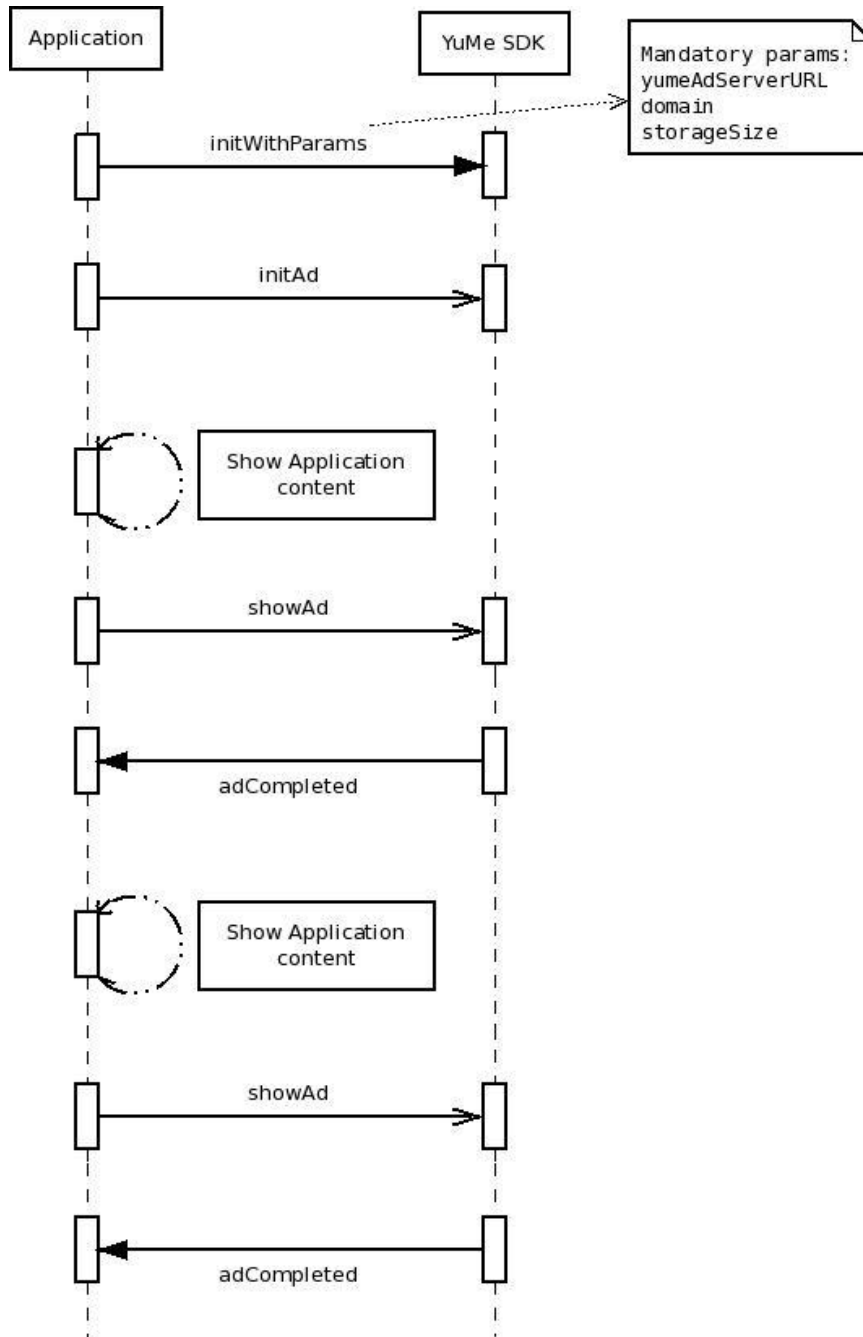
- (void)adExpired:(YuMeAdTypeEnum)adType {
    NSLog(@"Ad Expired");
}
```

Integration Support

Please email us at yume_support@yume.com or communicate directly with the integration specialist assigned if you run into integration issues.

Appendix

Plug-n-Play – API call sequence



Comprehensive Sample Code

Plug-n-Play Integration –

Sample code is also available in the SDK package

```

//
//  SampleViewController.h
//  YuMeSample
//

#import <UIKit/UIKit.h>
#import "YuMeSDK.h"
#import "YuMeSDKDelegate.h"

@interface SampleViewController : UIViewController <YuMeSDKDelegate> {
    YuMeSDK *yumeSDK;
}

- (void)initYuMeSDK;
- (void)initiateAdPreFetch;
- (void)displayAd;

@end

//
//  SampleViewController.m
//  YuMeSample
//

#import "SampleViewController.h"

@implementation SampleViewController

    // The designated initializer.  Override if you create the controller
    programmatically and want to perform customization that is not
    appropriate for viewDidLoad.

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle
*)nibBundleOrNil {
    if ((self = [super initWithNibName:nibNameOrNil
bundle:nibBundleOrNil])) {
        // Custom initialization
        [self initYuMeSDK];
    }
    return self;
}

- (void)dealloc {
    [yumeSDK release];
}

```



```

        [super dealloc];
    }

    /*
    Sample function to initialize the YuMe SDK.
    */
- (void)initYuMeSDK {
    NSError *err = nil;
    YuMeAdParams *params = [[YuMeAdParams alloc] init];

    params.domain = @"704oIaHzpGu";
    params.yumeAdServerURL = @"http://shadow01.yumenetworks.com/";
    params.additionalParams = @"";
    params.storageSize = 5;

    yumeSDK = [[YuMeSDK alloc] initWithParams:params
errorStatus:&err];
    [params release];

    if (!yumeSDK) {
        NSString *errString = [YuMeSDK getErrDesc:err];
        NSLog(@"YuMe SDK initialization failed: %@", errString);
    }
    else {
        [yumeSDK setDelegate:self];
        NSLog(@"YuMe SDK was initialized successfully");
    }
}

/*
Sample function to initiate pre-fetching of Ads in the background.
It needs to be called only once.
Ads will be automatically pre-fetched when required.
*/
- (void)initiateAdPreFetch {
    NSError *err = nil;
    BOOL b = [yumeSDK initAd:YuMeAdTypePreRoll errorStatus:&err];

    if (!b) {
        NSString *errString = [YuMeSDK getErrDesc:err];
        NSLog(@"initAd failed: %@", errString);
    }
    else {
        NSLog(@"initAd returned successfully");
    }
}

```

```

}

/*
Sample function to display an Ad that was pre-fetched.
*/
- (void)displayAd {
    YuMeAdTypeEnum adType = YuMeAdTypePreRoll;

    NSError *err = nil;

    NSLog(@"Ad available for: %d", adType);

    // If needed, replace self.view with application created
    view where the Ad needs to be displayed

    [yumeSDK setParentView:self.view];
    BOOL b = [yumeSDK showAd:adType errorStatus:&err];

    if (!b) {
        NSString *errString = [YuMeSDK getErrDesc:err];
        NSLog(@"showAd failed: %@", errString);
        // Application continues with content
    }
    else {
        NSLog(@"showAd returned successfully");
        // Application waits for adCompleted event
    }
}

}

/*
///// YuMe SDK's event handling start /////
*/

- (void)adPresent {
    NSLog(@"Ad present");
}

- (void)adAbsent {
    NSLog(@"Ad absent");
}

- (void)adPlaying {
    NSLog(@"Ad playing");
}

```

```

- (void)adCompleted {
    NSLog(@"Ad completed");
    // Application continues with content
}

- (void)adError:(NSError *)error {
    NSLog(@"Ad error");
}

- (void)adReady:(YuMeAdTypeEnum)adType assetsCached:(BOOL)isCached {
    NSLog(@"Ad ready: %@", (isCached ? @"TRUE" : @"FALSE"));
}

- (void)adNotReady:(YuMeAdTypeEnum)adType {
    NSLog(@"Ad not ready");
}

- (void)adExpired:(YuMeAdTypeEnum)adType
{
    NSLog(@"Ad Expired");
    NSError *err = nil;
    BOOL b = [yumeSDK initAd:YuMeAdTypePreRoll errorStatus:&err];
    if (!b) {
        NSString *errString = [YuMeSDK getErrDesc:err];
        NSLog(@"initAd failed: %@", errString);
    }
    else {
        NSLog(@"Prefetching pre-roll Ad");
    }
}

/*
///// YuMe SDK's event handling end /////
*/

/*
Uncomment the following if your application supports orientation
change.

// Override to allow orientations other than the default portrait
orientation.
-
(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)i
nterfaceOrientation {
    return YES;
}

```

```
}  
  
-  
(void)didRotateFromInterfaceOrientation:(UIInterfaceOrientation)fromIn  
terfaceOrientation {  
    // Make changes to application UI first (if required) ad then  
    call the following API.  
    [yumeSDK orientationChanged];  
}  
*/  
@end
```

Apple®, iPhone®, and iPad® are registered trademarks of Apple, Inc.

App StoreSM is a service mark of Apple, Inc.