

# Workspot iOS SDK Reference Document

Version 1

# Contents

<b>1</b>	<b>Workspot iOS SDK</b>	<b>3</b>
1.1	Main Page	3
1.2	Class Documentation	12
1.2.1	WorkspotSDK Class Reference	12
1.2.1.1	Detailed Description	12
1.2.1.2	Method Documentation	12
1.2.1.3	Property Documentation	12
1.2.2	WSAnalytics Class Reference	13
1.2.2.1	Detailed Description	13
1.2.2.2	Method Documentation	14
1.3	File Documentation	14
1.3.1	WorkspotSDK.h File Reference	14
1.3.1.1	Typedef Documentation	14
1.3.2	WSAnalytics.h File Reference	14
1.3.2.1	Macro Definition Documentation	14
	<b>Index</b>	<b>15</b>



# Chapter 1

## Workspot iOS SDK

### 1.1 Main Page

#### Initialization

Developers do not need to call any initialization methods. SDK initialization is done on client launch. Just include the following header file

```
#import <WorkspotSDK/WorkspotSDK.h>
```

#### Auditing Events

Use the following macro to send audit events.

```
WSANALYTICS_EVENT(@"John Doe looked up more info on %@", @"Jane Doe");
```

#### Code Samples

##### Example with no customization

```
#import <WorkspotSDK/WorkspotSDK.h>

- (void) AcmeAppUserDidPrintDocument:(NSString*)documentName
                                   pageNum:(NSString*)pageNum
{
    WSANALYTICS_EVENT(@"User printed page number %@ of document %@", pageNum, documentName);
}
```

##### Example for troubleshooting use-case

```
#import <WorkspotSDK/WorkspotSDK.h>

- (BOOL) AcmeAppApplication:(UIApplication *)application
willFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    [WorkspotSDK sharedInstance].debugLogEnabled = YES;
}

- (void) AcmeAppUserDidPrintDocument:(NSString*)documentName
                                   pageNum:(NSString*)pageNum
{
    WSANALYTICS_EVENT(@"User printed page number %@ of document %@", pageNum, documentName);
}
```

##### Example for troubleshooting use-case with Custom Logging

```
#import <WorkspotSDK/WorkspotSDK.h>

- (BOOL) AcmeAppApplication:(UIApplication *)application
willFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    [WorkspotSDK sharedInstance].debugLogEnabled = YES;
    [WorkspotSDK sharedInstance].errorLogEnabled = YES;
    [WorkspotSDK sharedInstance].debugLogBlock = ^(NSString *fmt, ...)
    {
        /*
         * Custom Debug Log callback
         */
        va_list argList;
        va_start(argList, fmt);
        NSLogv(fmt, argList);
        va_end(argList);
    };

    [WorkspotSDK sharedInstance].errorLogBlock = ^(NSString *fmt, ...)
    {
        /*
         * Custom Error Log callback
         */
        va_list argList;
        va_start(argList, fmt);
        NSString *s = [[NSString alloc] initWithFormat:fmt arguments:argList];
        NSLog(@"ERR: %@", s);
        va_end(argList);
    };
}

- (void) AcmeAppUserDidPrintDocument:(NSString*)documentName
                                pageNum:(NSString*)pageNum
{
    WSANALYTICS_EVENT(@"User printed page number %@ of document %@", pageNum, documentName);
}
```

Please note that you should be able to use `iOS Simulator` to successfully build and link your new application but, you need to create and upload an IPA (see below) to `Workspot Control` in order to test end to end functionality.

## Xcode Integration and Settings

Workspot iOS SDK requires iOS 7 and Xcode 5.0.2.

You would need to drag and drop `WorkspotSDK.framework` to your Xcode Project. We recommend you copy the framework file into your project's repository as shown

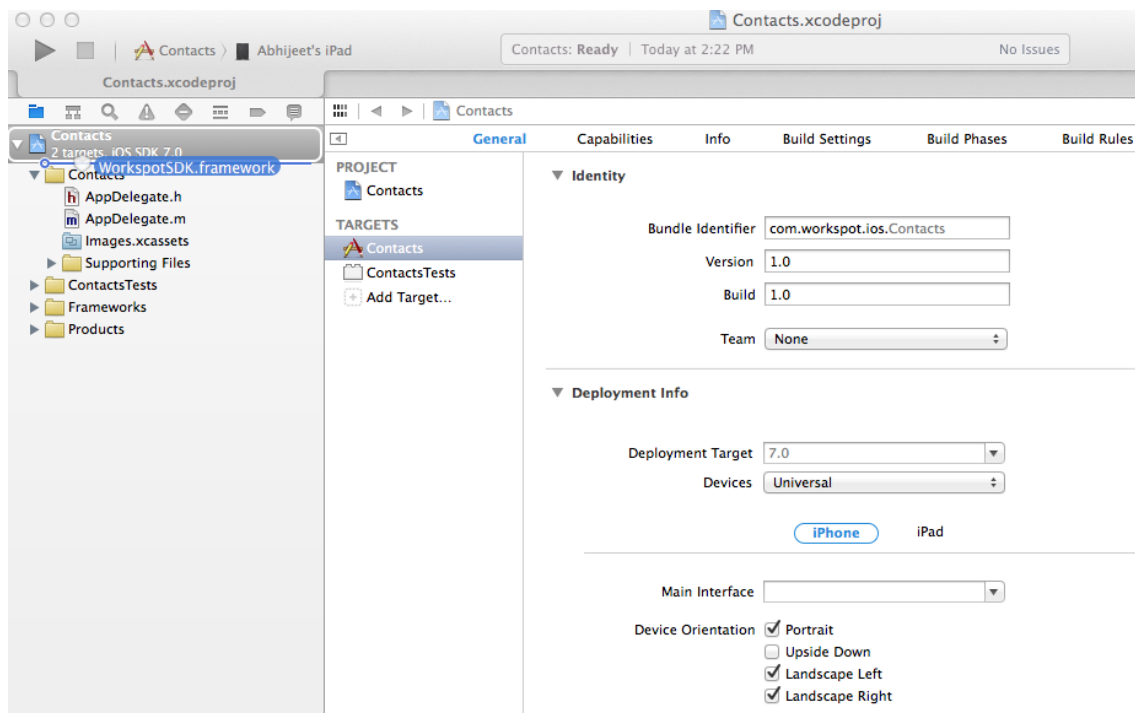


Figure 1.1: Adding Framework

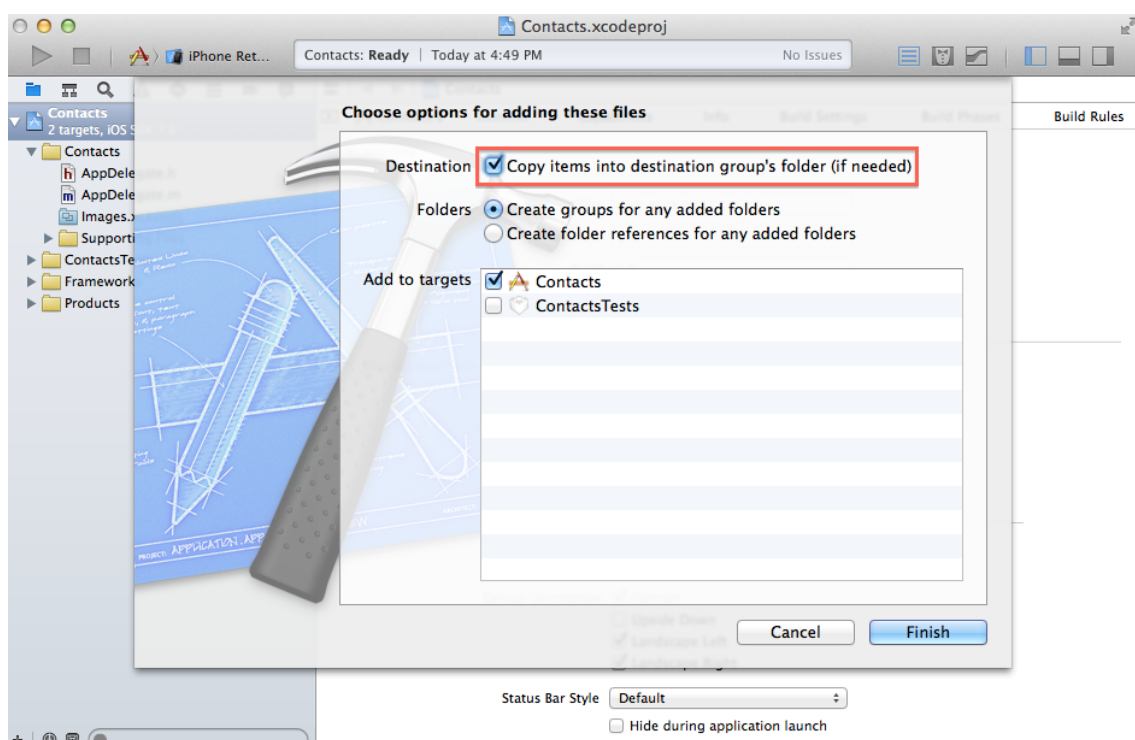


Figure 1.2: Copy Framework

After you have copied the framework, you should see `WorkspotSDK.framework` in your Xcode's Project Navigator as shown

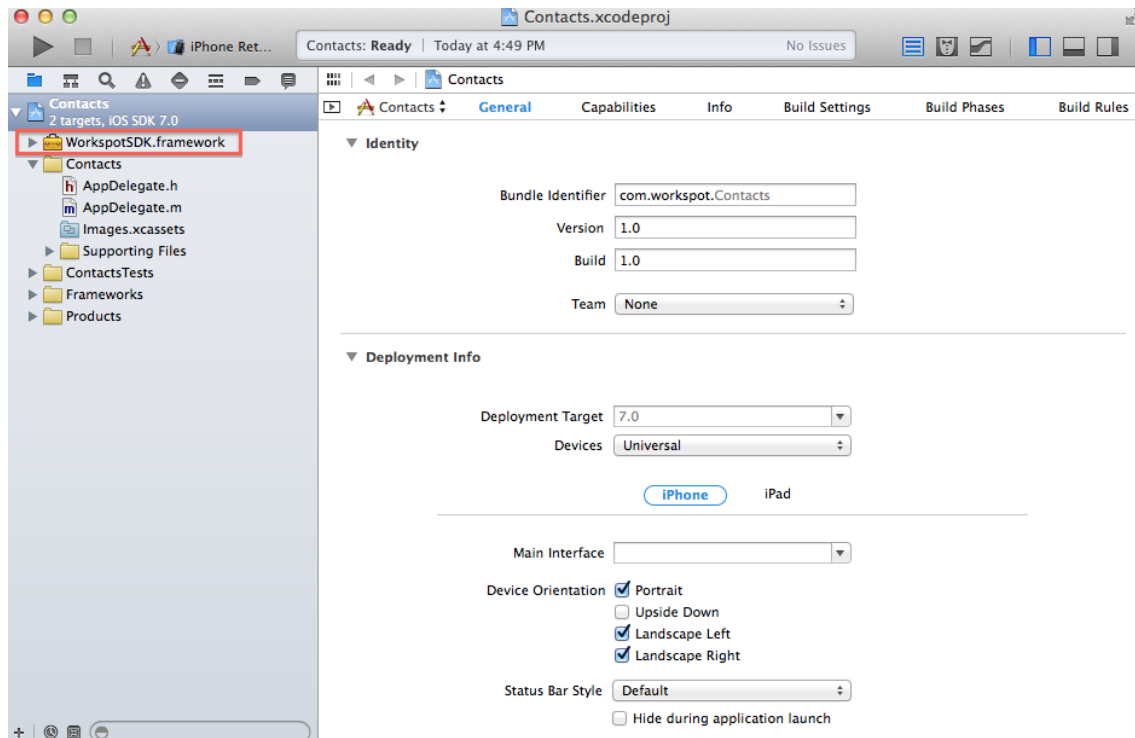


Figure 1.3: Copied Framework

iOS Workspot SDK requires you to include the following frameworks in your Xcode Build Phases:

- `SystemConfiguration.framework`
- `CoreLocation.framework`
- `CoreTelephony.framework`

`WorkspotSDK.framework` is dependent on these three frameworks in addition to `Foundation.framework`. If you already have these linkages, you don't need to re-add them again.

Alternatively, you can always use the new `@import` feature in Xcode 5.

```
#import <AppDelegate.h>

#import SystemConfiguration
#import CoreLocation
#import CoreTelephony
```

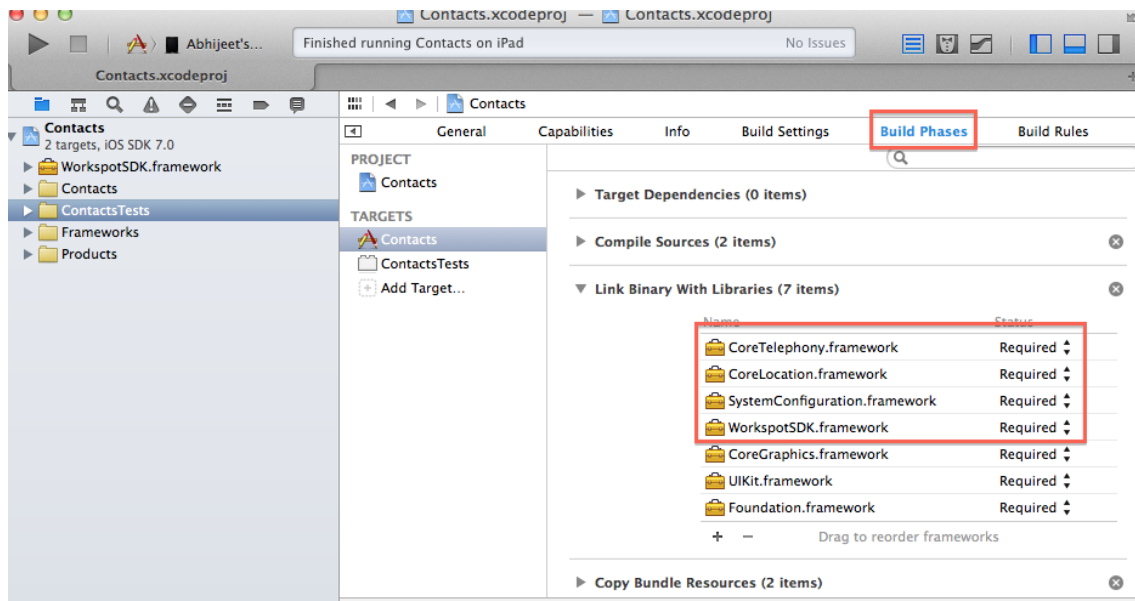


Figure 1.4: Framework Includes

iOS Workspot SDK uses Objective-C Categories and hence, requires your Build Settings to have `-ObjC` added to your Other Linker Flags.

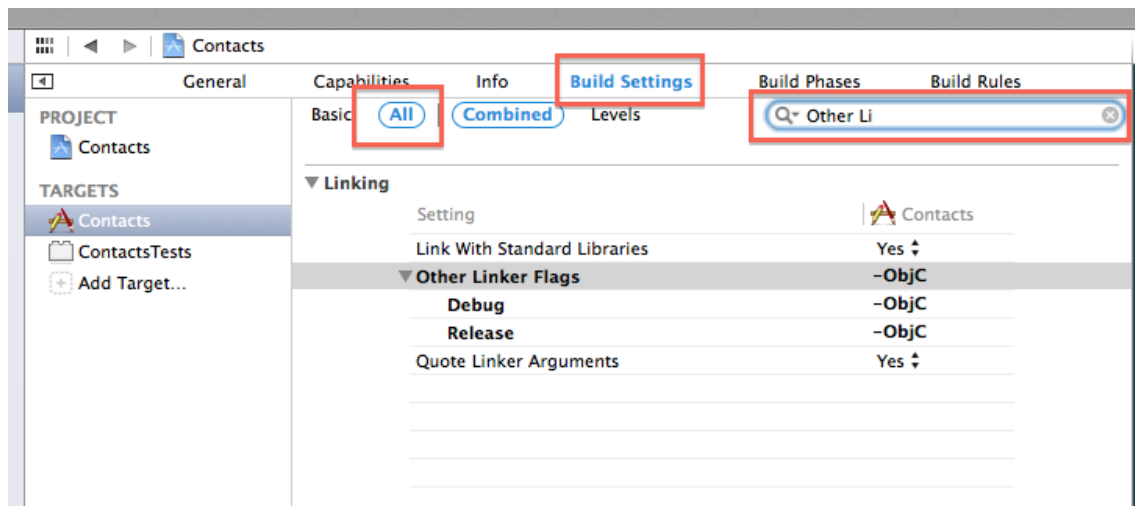


Figure 1.5: Other Linker Flags

## Uploading IPA

After you built your IPA file, you will need to upload the application on *Workspot Control* as shown in the following figure.



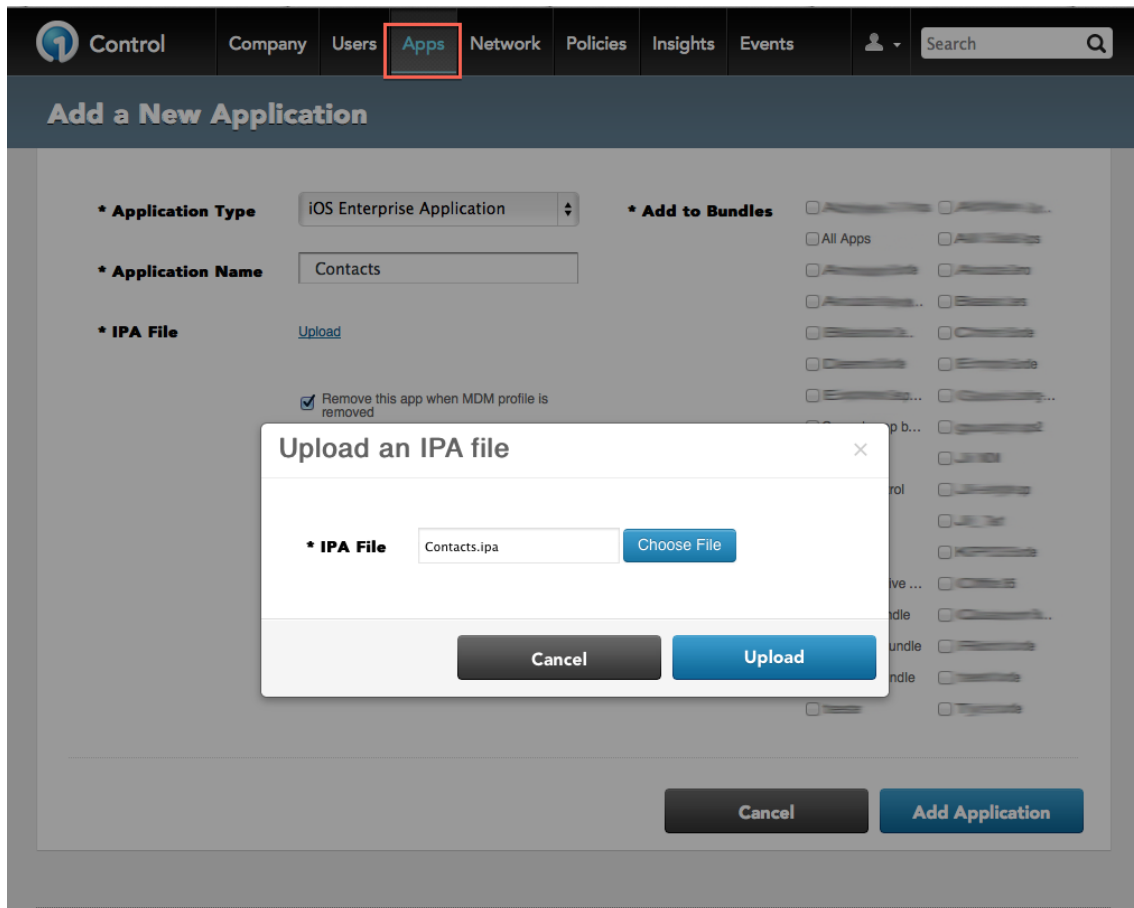


Figure 1.6: Upload IPA

Please make sure you select the *SDK* flag before you "Save Application". This will activate *Workspot SDK* in your application and start uploading analytics data.

Control

Company

Users



Apps

Network

Policies

Insights

Events

 Search 

## Add a New Application

\* Application Type

iOS Enterprise Application

\* Application Name

Contacts

\* IPA File

[Upload](#)

☒ Remove this app when MDM profile is removed

☒ Do not backup data generated by this app when the device is synced

☒ This application uses workspot SDK

\* Add to Bundles

☐ All Apps

☐

☐

☐

☐

☐

☐

☐

☐

☐

☐

☐

☐

☐

☐

☐

☐

☐

☐

☐

☐

Cancel

Add Application

Figure 1.7: SDK Flag

After you have assigned the application to a user bundle and successfully installed the application on your device, the events will be accessible on the *Events* tab of *Workspot Control*.

Control

Company

Users

Apps

Network

Policies

Insights

Events

## Events

Hide

From

To

Narrow By

Applications

☒ All Apps
 ☐ Select Apps

Geos

☒ All Geos
 ☐ Select Geos

Networks

☒ All Networks

Apply

Date & Time	Event	User Name	Geo	Device
11:36 AM 02/11/2014	[REDACTED]	abhijeet	California	iPad Mini, WiFi
11:36 AM 02/11/2014	John Doe looked up more info on Jane Doe	abhijeet	California	iPad Mini, WiFi
9:37 AM 02/11/2014	[REDACTED]	abhijeet	California	iPad Mini, WiFi
9:37 AM 02/11/2014	[REDACTED]	abhijeet	California	iPad Mini, WiFi
9:37 AM 02/11/2014	[REDACTED]	abhijeet	California	iPad Mini, WiFi
9:37 AM 02/11/2014	[REDACTED]	abhijeet	California	iPad Mini, WiFi
9:37 AM 02/11/2014	[REDACTED]	abhijeet	California	iPad Mini, WiFi
9:36 AM 02/11/2014	[REDACTED]	abhijeet	California	iPad Mini, WiFi
9:36 AM 02/11/2014	[REDACTED]	abhijeet	California	iPad Mini, WiFi

Figure 1.8: Control Events

## Troubleshoot

### Build Time Troubleshooting

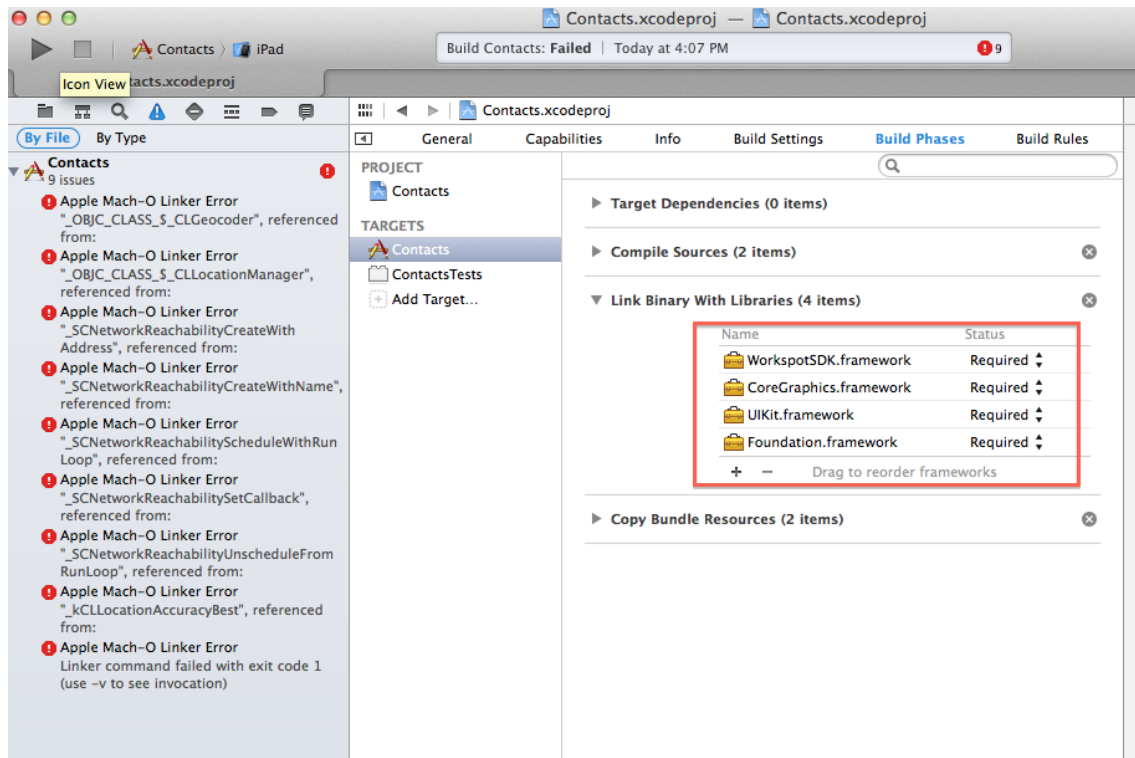


Figure 1.9: Forgot Frameworks

If you run into build time errors similar to above, please check if you have the following frameworks linked

- `SystemConfiguration.framework`
- `CoreLocation.framework`
- `CoreTelephony.framework`

## Runtime Troubleshooting

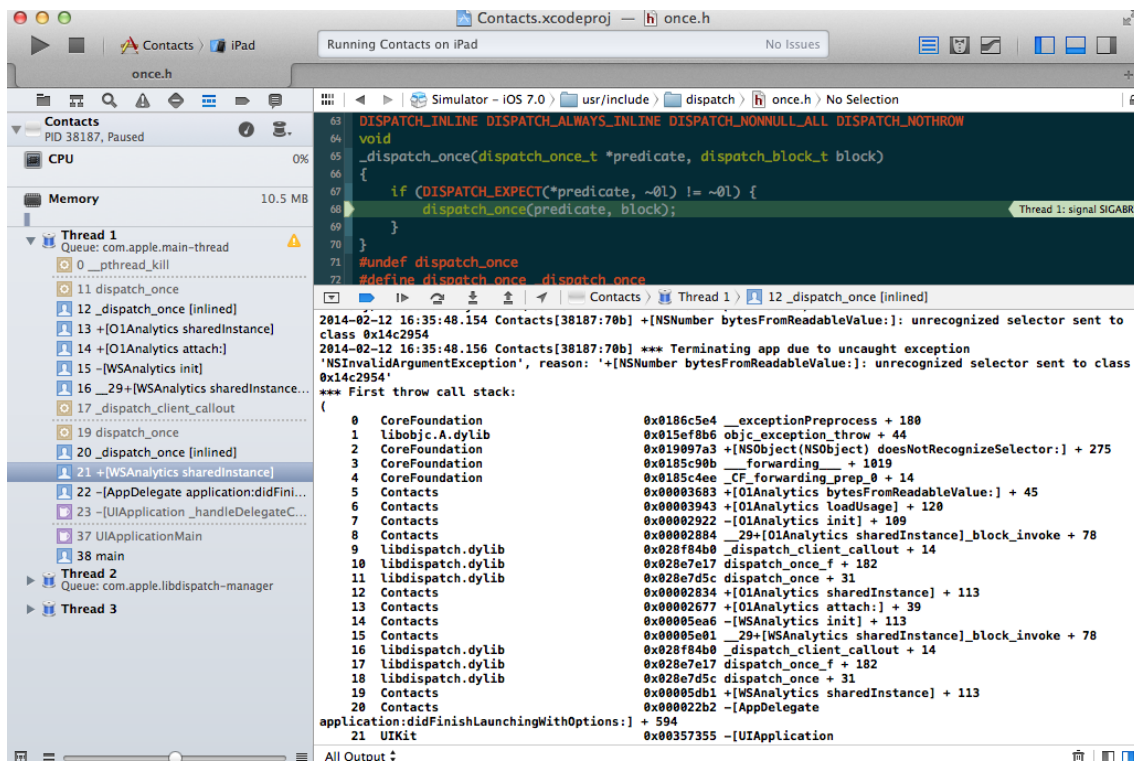


Figure 1.10: Forgot ObjC

If you run into a run time error immediately on launch similar to above, please make sure you have `-ObjC` flag set for Other Linker Flags in Build Settings.

## Events Missing

To troubleshoot events missing on Workspot Control, developers can optionally enable logs and re-run their application. Debug logs are not enabled by default and can be turned on for troubleshooting. Please send these logs to [support@workspot.com](mailto:support@workspot.com) so we can better assist you in debugging.

```
[WorkspotSDK sharedInstance].debugLogEnabled = YES;
```

Errors are logged by default. Developers have an option to turn it off using

```
[WorkspotSDK sharedInstance].errorLogEnabled = NO;
```

Developers can override default `NSLog(...)` by registering to the following debug callback

```
[WorkspotSDK sharedInstance].debugLogBlock = ^(NSString *fmt, ...)
{
    /*
     * Custom Debug Log callback
     */
};
```

and following for error callback

```
[WorkspotSDK sharedInstance].errorLogBlock = ^(NSString *fmt, ...)
{
    /*
     * Custom Error Log callback
     */
};
```

## 1.2 Class Documentation

### 1.2.1 WorkspotSDK Class Reference

General entry point where you do necessary initializations.

Inherits NSObject.

#### Class Methods

- (instancetype) + [sharedInstance](#)  
*Shared Instance for [WorkspotSDK](#).*

#### Properties

- BOOL [debugLogEnabled](#)  
*Knob to enable or disable [WorkspotSDK](#) debug logs.*
- BOOL [errorLogEnabled](#)  
*Knob to enable or disable [WorkspotSDK](#) error logs.*
- [WSCustomLogBlock](#) [debugLogBlock](#)  
*Block for custom debug logs.*
- [WSCustomLogBlock](#) [errorLogBlock](#)  
*Block for custom error logs.*

#### 1.2.1.1 Detailed Description

General entry point where you do necessary initializations.

#### 1.2.1.2 Method Documentation

##### 1.2.1.2.1 + (instancetype) sharedInstance

Shared Instance for [WorkspotSDK](#).

#### 1.2.1.3 Property Documentation

##### 1.2.1.3.1 -(WSCustomLogBlock) debugLogBlock [read],[write],[nonatomic],[strong]

Block for custom debug logs.

Once this block is set, you will receive callback notifications when SDK tries to log debug information. You can print this information to the console or log it into a file. This information will be needed by Workspot development team when troubleshooting Workspot SDK.

#### Note

This callback will happen on a background thread.

##### 1.2.1.3.2 -(BOOL) debugLogEnabled [read],[write],[nonatomic],[assign]

Knob to enable or disable [WorkspotSDK](#) debug logs.

It is disabled by default.

If this property is set, these logs are outputted to console using `NSLog ( . . . )` by default. You can override this by registering your own callback in property [debugLogBlock](#)

---

**Note**

Enable it only for [WorkspotSDK](#) troubleshooting purposes.

**1.2.1.3.3 - (WSCustomLogBlock) errorLogBlock** [read], [write], [nonatomic], [strong]

Block for custom error logs.

Once this block is set, you will receive callback notifications when SDK tries to log information for error scenarios. You can print this information to the console or log it into a file. This information will be needed by Workspot development team when troubleshooting Workspot SDK.

**Note**

This callback will happen on a background thread.

**1.2.1.3.4 - (BOOL) errorLogEnabled** [read], [write], [nonatomic], [assign]

Knob to enable or disable [WorkspotSDK](#) error logs.

It is enabled by default.

These logs are outputted to console using `NSLog ( . . . )` by default. You can override this by registering your own callback in property [errorLogBlock](#)

**Note**

Please contact us at [support@workspot.com](mailto:support@workspot.com) should you continue to see error logs coming from Workspot SDK.

**1.2.2 WSAalytics Class Reference**

Class for auditing Events data.

Inherits NSObject.

**Instance Methods**

- (void) - [auditWithFormat:](#)

*Audits Events data.*

**Class Methods**

- (instancetype) + [sharedInstance](#)

*Singleton instance of [WSAalytics](#).*

**1.2.2.1 Detailed Description**

Class for auditing Events data.

The recommended approach to audit events is using macro [WSANALYTICS\\_EVENT](#)

**Examples**

```
WSANALYTICS_EVENT(@"John Doe looked up more info on %@", @"Jane Doe");
```

---

### 1.2.2.2 Method Documentation

#### 1.2.2.2.1 - (void) auditWithFormat: (NSString \*) *format*, ...

Audits Events data.

##### Parameters

<i>format</i>	Format for variable argument list
...	Variable argument list

#### 1.2.2.2.2 + (instancetype) sharedInstance

Singleton instance of [WSAnalytics](#).

## 1.3 File Documentation

### 1.3.1 WorkspotSDK.h File Reference

#### Classes

- class [WorkspotSDK](#)  
*General entry point where you do necessary initializations.*

#### Typedefs

- typedef void(^ [WSCustomLogBlock](#) )(NSString \*fmt,...)  
*Type definition for custom logging callbacks.*

#### 1.3.1.1 Typedef Documentation

##### 1.3.1.1.1 WSCustomLogBlock

Type definition for custom logging callbacks.

### 1.3.2 WSAalytics.h File Reference

#### Classes

- class [WSAnalytics](#)  
*Class for auditing Events data.*

#### Macros

- #define [WSANALYTICS\\_EVENT](#)(fmt,...)  
*Macro to audit events data.*

#### 1.3.2.1 Macro Definition Documentation

##### 1.3.2.1.1 #define WSANALYTICS\_EVENT( *fmt*, ... )

**Value:**

---

```
do
{
    [[WSAnalytics sharedInstance] auditWithFormat:fmt, ##__VA_ARGS__];
} while (0)
```

Macro to audit events data.

Use this macro to audit events in free form text



# Index

- auditWithFormat:
  - WSAnalytics, [14](#)
- debugLogBlock
  - WorkspotSDK, [12](#)
- debugLogEnabled
  - WorkspotSDK, [12](#)
- errorLogBlock
  - WorkspotSDK, [13](#)
- errorLogEnabled
  - WorkspotSDK, [13](#)
- sharedInstance
  - WorkspotSDK, [12](#)
  - WSAnalytics, [14](#)
- WSANALYTICS\_EVENT
  - WSAnalytics.h, [14](#)
- WSAnalytics, [13](#)
  - auditWithFormat:, [14](#)
  - sharedInstance, [14](#)
- WSAnalytics.h, [14](#)
  - WSANALYTICS\_EVENT, [14](#)
- WSCustomLogBlock
  - WorkspotSDK.h, [14](#)
- WorkspotSDK, [12](#)
  - debugLogBlock, [12](#)
  - debugLogEnabled, [12](#)
  - errorLogBlock, [13](#)
  - errorLogEnabled, [13](#)
  - sharedInstance, [12](#)
- WorkspotSDK.h, [14](#)
  - WSCustomLogBlock, [14](#)