Getting Started with Apple® iOS Development – Link-OS™ SDK – Objective-C

Overview

This document describes the end to end process of designing, packaging, deploying and running an Apple® iPhone /iPod application using the Link-OS SDK.

The sample code used in this guide is from the Connect and Print Over TCP/IP with Apple® iOS – Link-OS™ SDK – Objective-C sample code article.

Also included are sections about Registering (White Listing) Your App with Zebra and a Grand Central Dispatch (GCD) Overview.

Target Audience

The information delivered in this document assumes the reader has reasonable technical competence covering Apple®iOS environment, core programming concepts and rationales.

Feedback

We value all feedback. Please rate the article and then add your comments/suggestions in the text field that appears.

System Prerequisites

You must install all the system prerequisites listed in this section.

Note: The default installation options for all these prerequisites are satisfactory.

MAC OS X

Unlike development for other mobile device environments, development for Apple®iOS based mobile devices must be conducted on Mac OS X.

Refer to Mac OS X.

Xcode and iOS SDK

Xcode is the development environment for apps that run on the Apple® iOS operating system.

For more information and to download links, refer to for Xcode and the iOS SDK.

Zebra Multiplatform SDK

The Link-OS iOS digital devices SDK contains all the required components to develop applications for Zebra label printers. The SDK includes the header files to scan for and connect to network based Zebra label printers.

For more information, including system prerequisites and download links for Link-OS SDK,

Apple® iOS Device

While the Apple® iOS simulator included in the Xcode toolkit satisfies most of the anticipated requirements for developing with Apple® iOS, we recommend that you test all of your development with a physical device.

Registering (White Listing) Your App with Zebra

If your App will be posted to the Apple AppStore^{sm} and you plan to connect an iOS device to a Zebra MFi printer over a Bluetooth connection, Apple requires you to register your app with Zebra.

For more information about this process and requirements, read iOS App White Listing Frequently Asked Questions,

Grand Central Dispatch (GCD) Overview

Grand Central Dispatch is a feature that consists of language features, runtime libraries and system enhancements that provide methodical and inclusive alterations to the support of runtime code execution on multicore hardware in iOS.

The Cocoa Touch API has been improved to use GCD in order to assist the device and app to run quicker and more efficiently. GCD operates at the system level for an app and makes multitasking feasible via the use of multithreading. It can allocate resources for one thread while being able to process another.

Examples in this document dispatch the GUI portion of the application while trying to establish a connection to a printer. Rather than using a sequential and iterative approach to run tasks, GCD simplifies the app for both the developer and user to respectively create and use an app that works smoothly with no hesitation between tasks.

Note: After the release of iOS 6.1, Apple requires that developers reference APIs using GCD.

Creating an Apple® iPhone /iPod application using the Link-OS SDK

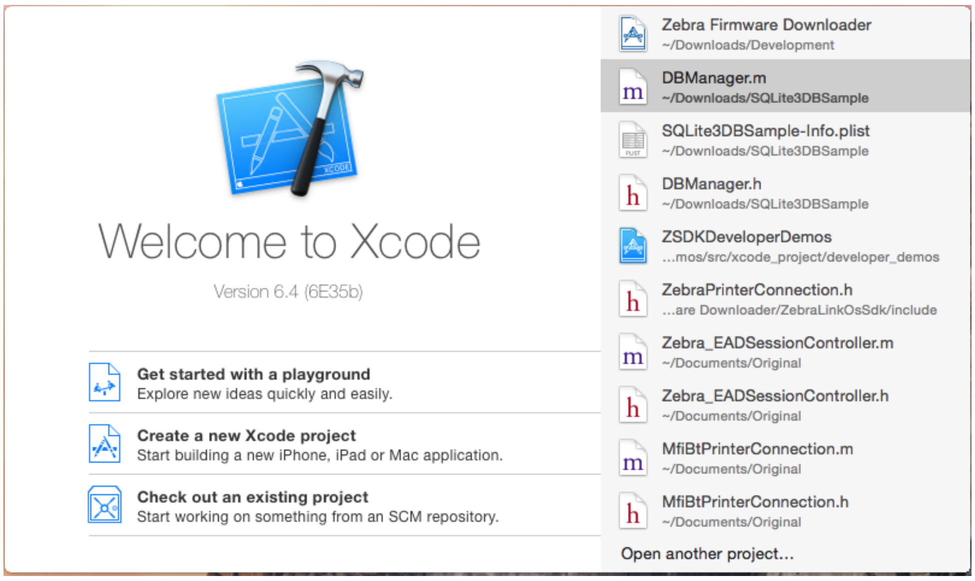
You will be guided on how to create a mobile application in the following sections:

- Designing a New Xcode Project
- Building the User Interface
- Linking the User Interface to Your Project
- Configuring Your Project to Include the Link-OS SDK
- Building and Running Your Project

Designing a New Xcode Project

1. Start Xcode.

   The Xcode welcome window appears.



Welcome to Xcode

Version 6.4 (6E35b)

Get started with a playground
Explore new ideas quickly and easily.

Create a new Xcode project
Start building a new iPhone, iPad or Mac application.

Check out an existing project
Start working on something from an SCM repository.

Zebra Firmware Downloader
~/Downloads/Development

DBManager.m
~/Downloads/SQLite3DBSample

SQLite3DBSample-Info.plist
~/Downloads/SQLite3DBSample

DBManager.h
~/Downloads/SQLite3DBSample

ZSDKDeveloperDemos
...mos/src/xcode_project/developer_demos

ZebraPrinterConnection.h
...are Downloader/ZebraLinkOsSdk/include

Zebra_EADSessionController.m
~/Documents/Original

Zebra_EADSessionController.h
~/Documents/Original

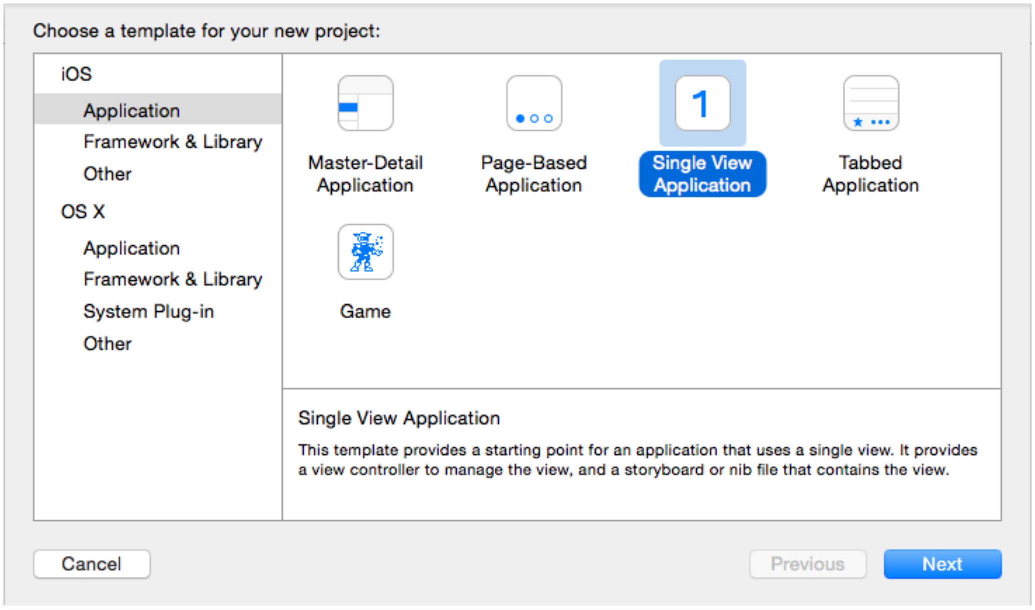
MfiBtPrinterConnection.m
~/Documents/Original

MfiBtPrinterConnection.h
~/Documents/Original

Open another project...

2. Click **Create a new Xcode project**.

   The template window appears.



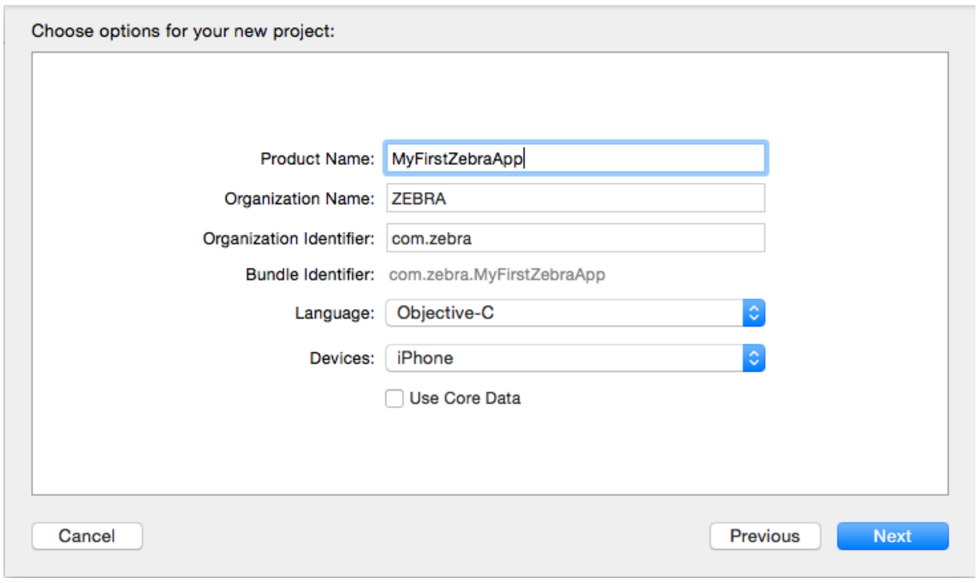
Choose a template for your new project:

Single View Application

This template provides a starting point for an application that uses a single view. It provides a view controller to manage the view, and a storyboard or nib file that contains the view.

3. Click **Single View Application** from the iOS Application section and click **Next**.

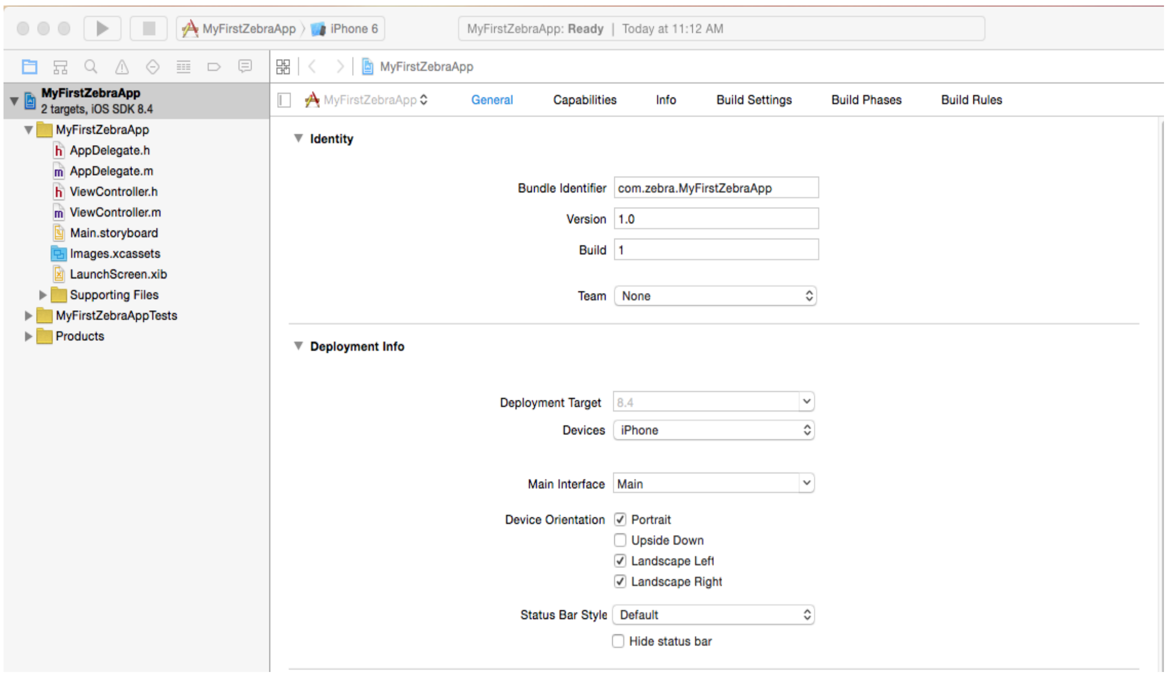
   A window appears allowing you to name your new project.


Choose options for your new project:

Product Name: MyFirstZebraApp
Organization Name: ZEBRA
Organization Identifier: com.zebra
Bundle Identifier: com.zebra.MyFirstZebraApp
Language: Objective-C
Devices: iPhone
☐ Use Core Data

4. Type “MyFirstZebraApp” as your project name and navigate to the location where you want to save your project.
5. Click **Next**.

   Xcode opens having prebuilt some source files for you.



Identity

Bundle Identifier: com.zebra.MyFirstZebraApp
Version: 1.0
Build: 1
Team: None

Deployment Info

Deployment Target: 8.4
Devices: iPhone
Main Interface: Main
Device Orientation: ☑ Portrait ☐ Upside Down ☑ Landscape Left ☑ Landscape Right
Status Bar Style: Default
☐ Hide status bar

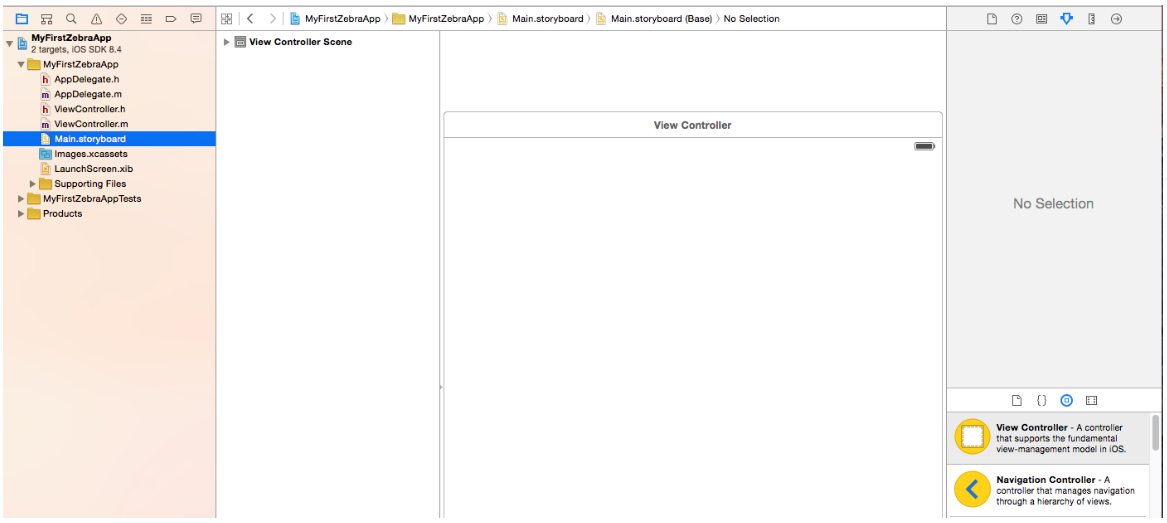
Building the User Interface

1. Locate and double click the “Main.storyboard” file.

   The application opens showing various windows including a:

   - Library of visual components
   - Blank design canvas

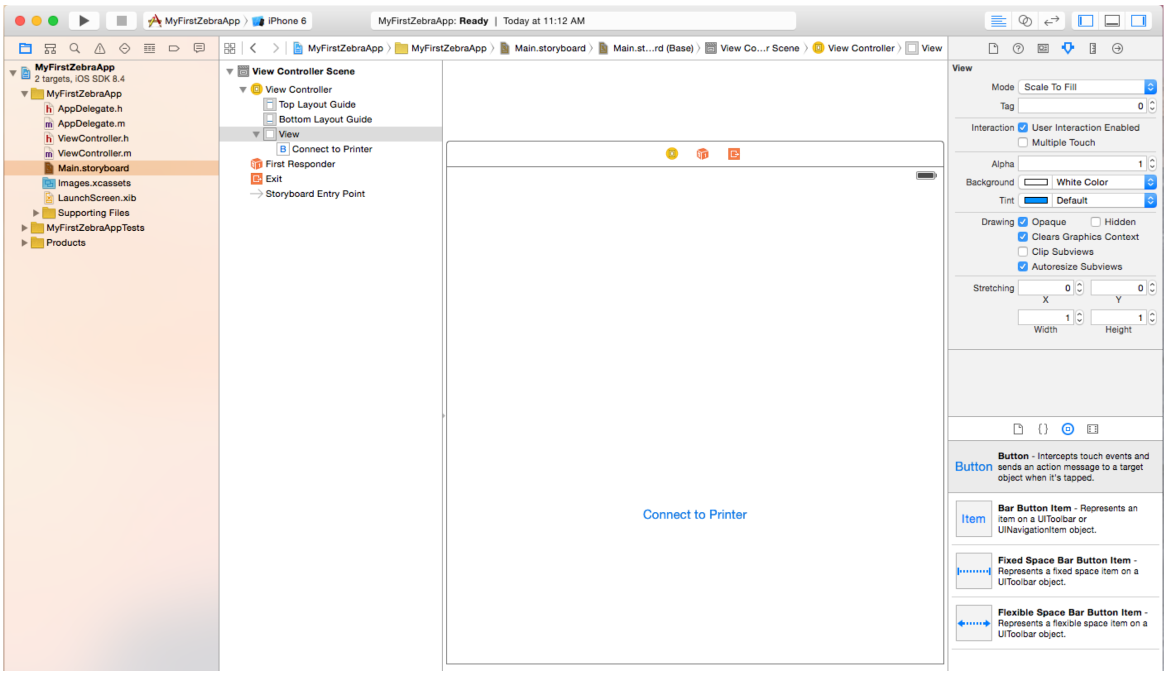
- Property window



View Controller - A controller that supports the fundamental view-management model in iOS.

Navigation Controller - A controller that manages navigation through a hierarchy of views.

2. Locate then drag and drop the **Button** item from the Library onto the blank View canvas.
3. Double click the button once it is on the canvas and title it "Connect to Printer".



View

Mode: Scale To Fill
Tag: 0
Interaction: User Interaction Enabled ✓; Multiple Touch
Alpha: 1
Background: White Color
Tint: Default
Drawing: Opaque ✓; Hidden; Clears Graphics Context ✓; Clip Subviews; Autoresize Subviews ✓
Stretching: X 0, Y 0, Width 1, Height 1

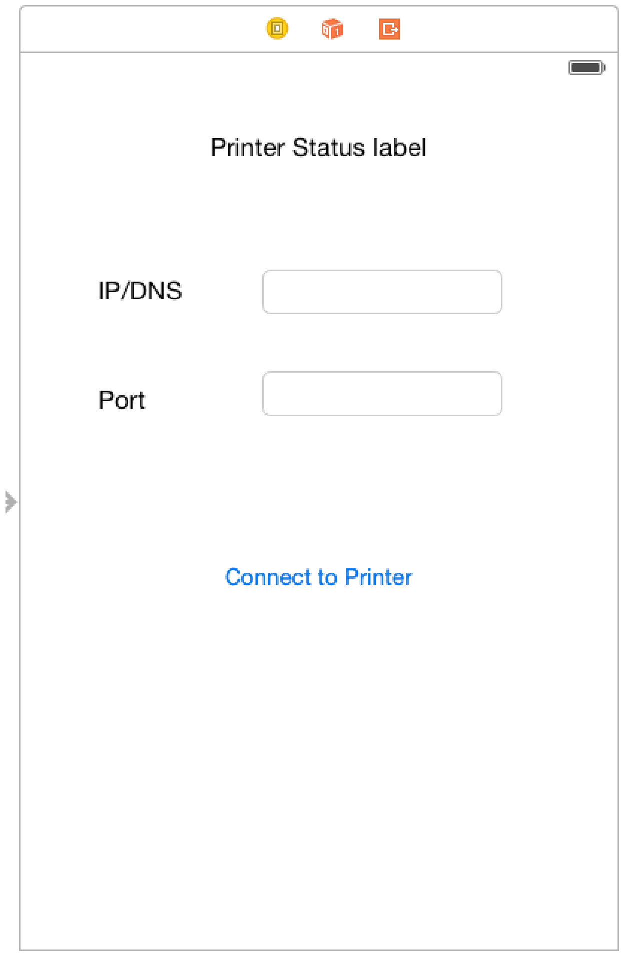
Button - Intercepts touch events and sends an action message to a target object when it's tapped.

Bar Button Item - Represents an item on a UIToolbar or UINavigationItem object.

Fixed Space Bar Button Item - Represents a fixed space item on a UIToolbar object.

Flexible Space Bar Button Item - Represents a flexible space item on a UIToolbar object.

4. Add 2 text fields and 3 label components to the canvas as the following dialog displays.



5. Close the Interface Builder and return to Xcode.

Linking the User Interface to Your Project

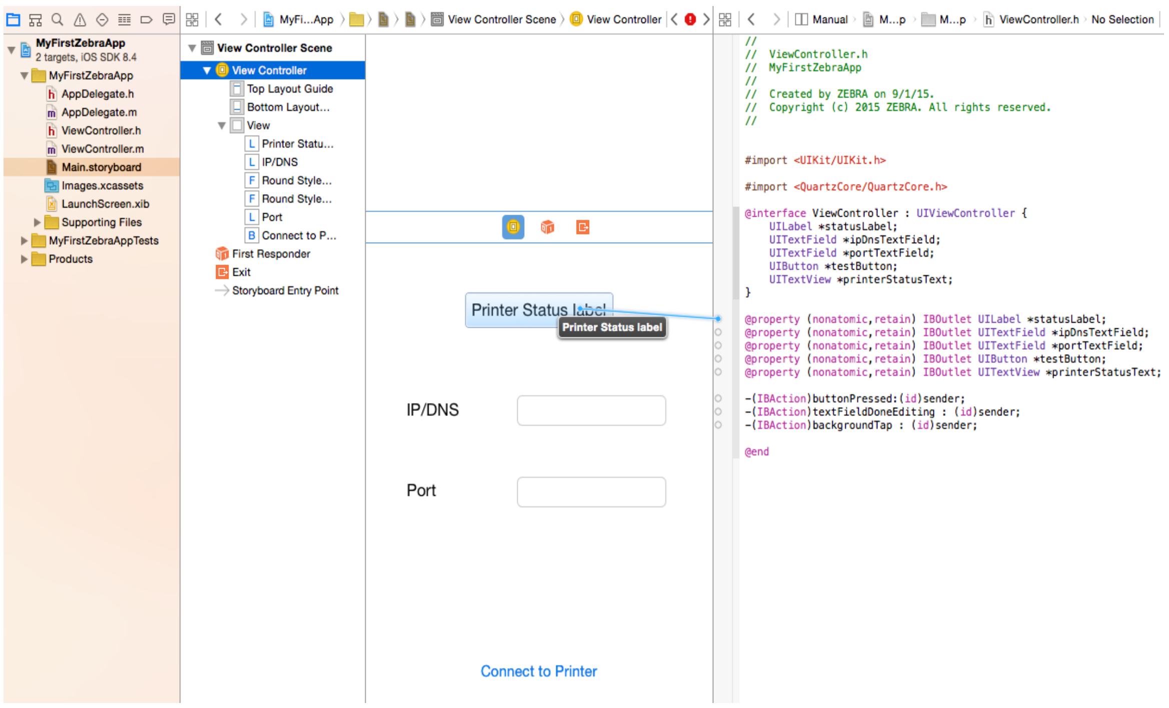
You must add the code to the main header file (ViewController.h) in order to support the components of the user interface.

1. Replace the code in the “ViewController.h” with the contents of the “MyFirstZebraAppViewController.h” file from the “Connection Over IP”. Make sure to update the class name.

   Note: The sample code used in this guide is from the Connect and Print Over TCP/IP with Apple® iOS – Link-OS™ SDK – Objective-C sample code article.

   For information on joining the ISV Program, contact a Program Manager in your area.

2. To now link the components of the user interface to the objects declared in the header file, open the “Main.storyboard” file again to open up the Interface.
3. Drag and drop the property status label from the ViewController.h to the printer status label in the “main. Storyboard”.



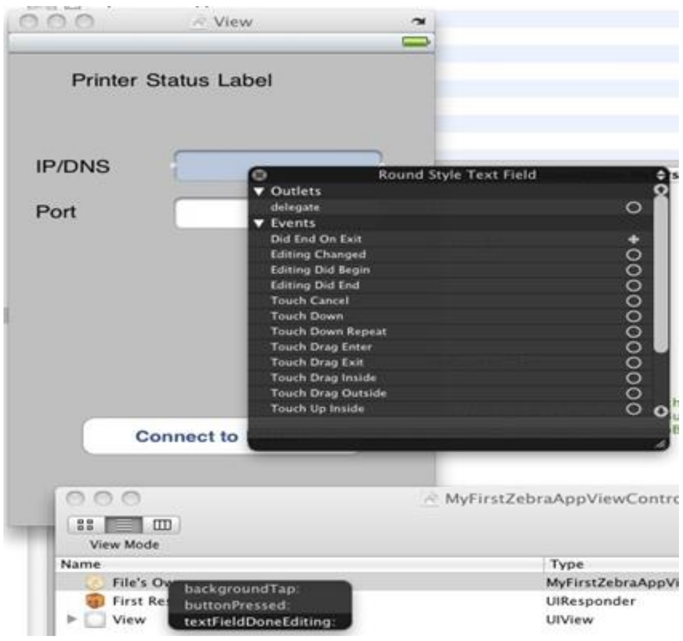
```
//
//  ViewController.h
//  MyFirstZebraApp
//
//  Created by ZEBRA on 9/1/15.
//  Copyright (c) 2015 ZEBRA. All rights reserved.
//


#import <UIKit/UIKit.h>

#import <QuartzCore/QuartzCore.h>

@interface ViewController : UIViewController {
    UILabel *statusLabel;
    UITextField *ipDnsTextField;
    UITextField *portTextField;
    UIButton *testButton;
    UITextView *printerStatusText;
}

@property (nonatomic,retain) IBOutlet UILabel *statusLabel;
@property (nonatomic,retain) IBOutlet UITextField *ipDnsTextField;
@property (nonatomic,retain) IBOutlet UITextField *portTextField;
@property (nonatomic,retain) IBOutlet UIButton *testButton;
@property (nonatomic,retain) IBOutlet UITextView *printerStatusText;

-(IBAction)buttonPressed:(id)sender;
-(IBAction)textFieldDoneEditing : (id)sender;
-(IBAction)backgroundTap : (id)sender;

@end
```

Repeat this process for the remaining properties.

Note: In the similar way, make sure the button pressed event [VeiwController.m] is linked to the “Connect to Printer” button on the user interface.

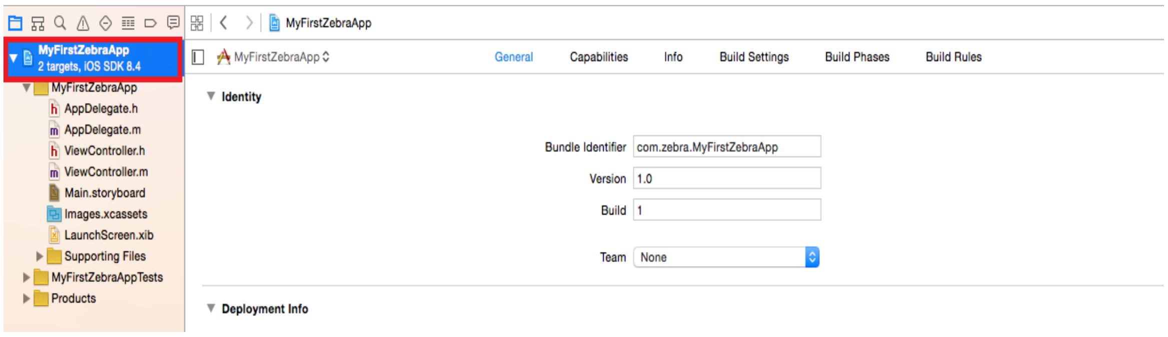
The text fields will be used to enter the IP/DNS name and port for printer. In order that the on-screen keyboard closes when the user finishes entering these settings, the “Did End On Exit” event for each text field must be handled.

4. Right click on the **IP/DNS** text box.
5. Drag the circle mark next to the “Did End On Exit” event onto “File’s Owner” and choose the “textFieldDoneEditing”.
6. Repeat this process for the Port text field.
7. Close the Interface Builder.



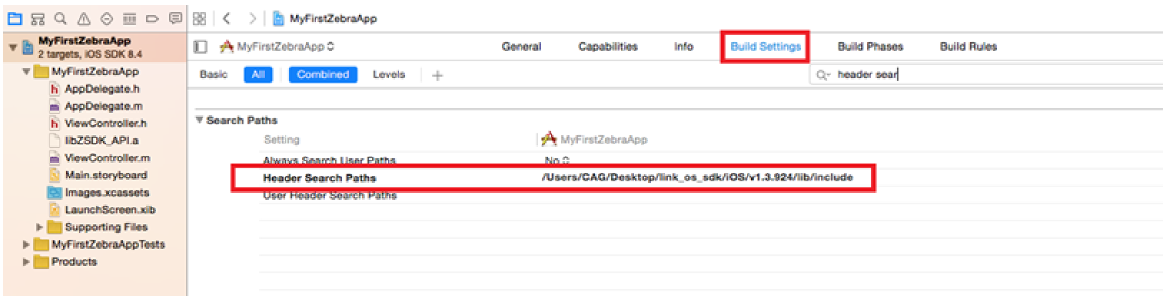
Configuring Your Project to Include the Link-OS SDK

1. Click the MyFirstZebraApp.
2. Click the Build Settings tab to specify the build information.



Bundle Identifier: com.zebra.MyFirstZebraApp
Version: 1.0
Build: 1
Team: None

3. Search for “Header Search Paths” by typing into the filter box at top of the window.



Search Paths

Setting	MyFirstZebraApp
Always Search User Paths	No
Header Search Paths	/Users/CAG/Desktop/link_os_sdk/iOS/v1.3.924/lib/include
User Header Search Paths	

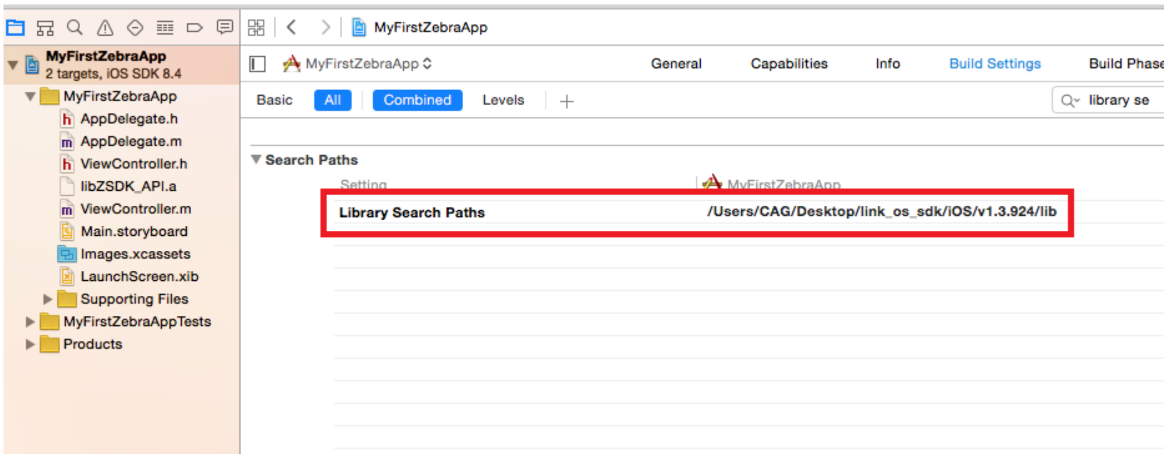
4. Edit the “Path” setting to link to the “include” folder in your installation path.

   For instance, if you have installed the Link-OS SDK to the default folder and your username is “macuser” then your path would be: /Users/macuser/Applications/zebralink_sdk/iOS/v1.3.924/lib/include

Note: DO NOT tick the “Recursive” option.

5. Click **OK**.

   You must now edit the “Library Search Paths”.



Library Search Paths /Users/CAG/Desktop/link_os_sdk/iOS/v1.3.924/lib

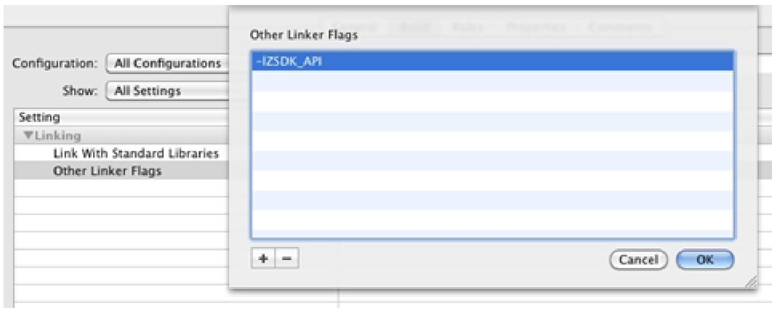
6. Edit the “Path” setting to link to the “lib” folder in your installation path.

   For instance, if you have installed the Link-OS SDK to the default folder and your username is “macuser” then your path would be:
   /Users/macuser/Applications/zebralink_sdk/iOS/v1.3.924/lib

   Note: DO NOT tick the “Recursive” option.

7. Click **OK**.

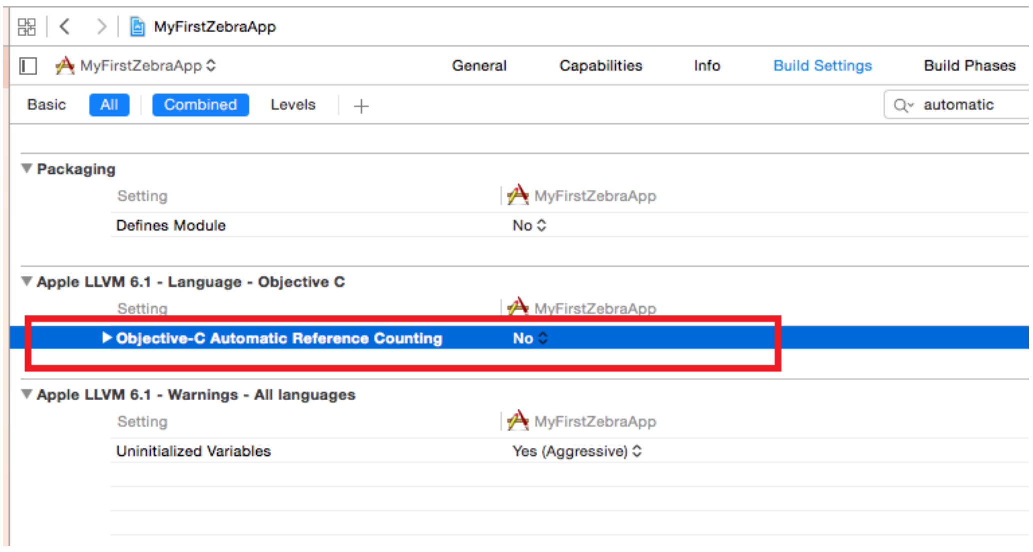
8. Locate the “Other Linker Flags” subheading and add the item “-LZSDK_API”.



-lZSDK_API

Note: You can choose to ignore this step if you directly drag and drop the libZSDK_API.a to the project folder.

You may need to disable the Automatic Reference Counting since you are using the implemented code.


MyFirstZebraApp

MyFirstZebraApp — General | Capabilities | Info | Build Settings | Build Phases

Basic | All | Combined | Levels — automatic

Packaging

Setting	MyFirstZebraApp
Defines Module	No

Apple LLVM 6.1 - Language - Objective C

Setting	MyFirstZebraApp
Objective-C Automatic Reference Counting	No

Apple LLVM 6.1 - Warnings - All languages

Setting	MyFirstZebraApp
Uninitialized Variables	Yes (Aggressive)

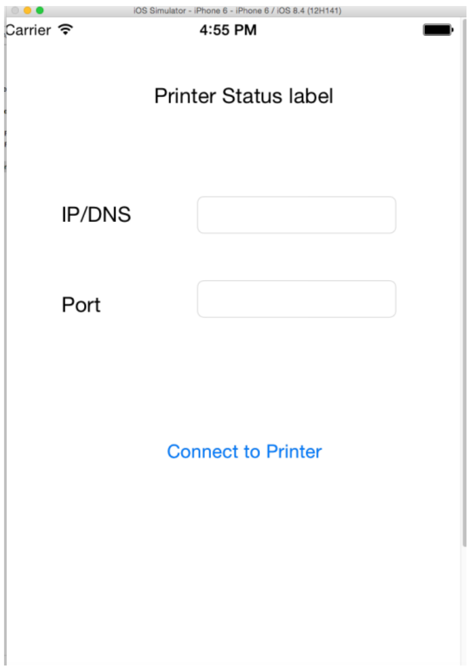
9. Replace the contents of the “ViewController.m” file with the source code from the “MyFirstZebraAppViewController.m” file from the “Connection Over IP”.

   Note: **Note:** The sample code used in this guide is from the Connect and Print Over TCP/IP with Apple® iOS – Link-OS™ SDK – Objective-C sample code article.

   With all the source code now added, you can build and run the solution.

Building and Running Your Project

Enter the IP/DNS address and Port number for the network printer and click the **Connect to Printer**.



A test label prints.

The “Printer Status Label” updates to reflect the connection and communication status with the printer.

Appendix

See Also

- Zebra SDK enabled Apple® iOS sample code and applications
- ZPL Programming Guide
- CPCL Programming Guide
- For any further information, sample code, application notes and solutions or to request further content, visit the Zebra Support Portal.

Document Control

Version	Date	Description
1	03-Dec-10	Initial Release
1.1	01-Mar-11	Minor updates to See Also section
2.0	28-Jun-12	Updated Chinese link into document
3.0	31-Jul-12	Updated Spanish and Portuguese link into document
4.0	01-Sep-15	Updated to match with latest iOS Xcode
5.0	11-Nov-15	Added the Registering (White Listing) Your App with Zebra and Grand Central Dispatch (GCD) Overview sections

Disclaimer