Developing an Apple® iOS Application for the iMZ – iOS Link-OS™ SDK – Objective-C

Overview

This document describes the major steps and tips to create a Bluetooth connecting iOS application with the iMZ using the Zebra's iOS Link-OS™ SDK.

Target Audience

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

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. This code has been developed to support iOS 6.1+.

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

Zebra Multiplatform SDK

The Zebralink for Apple® 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 Zebralink SDK,

Zebra iMZ Label Printer

In order to fully test the application created during the course of this tutorial, an iMZ printer should capable of understanding ZPL/LINE PRINT.

For more information, refer to Zebra iMZ label printers.

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 AppStoresm 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 requirement, refer to 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 whilst 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.

After release of iOS 6.1, Apple requires that developers reference APIs using GCD. If your Apple device runs iOS 6.1 or later then use below example to connect with iMZ printer otherwise you will not be able to connect.

Infinite Peripherals (IPC) Linea Pro Sled Overview

The Infinite Peripherals Linea Pro is a non-Zebra device created by Infinite Peripherals (IPC) that fits onto an iOS device and includes a barcode scanner and sometimes a card reader. Apple requires all apps that interact with a hardware peripheral device to be white listed by the manufacturer of that device (see Registering Your App with Zebra above). Apple identifies which peripheral devices are referenced through the protocol strings used. For example, to connect via Bluetooth to a Zebra iMZ printer the protocol com.zebra.rawport should be used in the code.

Because many of Zebra's ISVs who use iOS devices also use IPC devices Zebra included how to connect to an IPC device in the sample code in our SDK. Some ISVs include the protocols ("com.datecs.linea.pro.msr" and "com.datecs.linea.pro.bar") accidentally thinking it is necessary to connect to a Zebra printer. These protocols are only necessary if customers use the IPC sleds for scanning, card reading, or to facilitate Bluetooth pairing.

If the customers of the application to be white listed are purchasing and using the IPC sled then it should keep the IPC protocols in the code. However, because the IPC sled is a separate peripheral piece of hardware from the printer and has its own connection to the iOS device, Apple requires that the app be white listed with IPC as well. For more information, refer to Infinite Peripherals' Development Portal.

You must:

- Include the External Accessory framework in your project to be able to use this class.

- Include the IPC protocol strings "com.datecs.linea.pro.msr" and "com.datecs.linea.pro.bar" in your info.plist file under "Supported external accessory protocols". The “com.lineapro…etc”protocol essentially references the aforementioned.
- Include the IPC static library in your app for the class to work.
- Set the key "Required Background modes" to "App Communicates with an accessory" in your app's plist file

If the customers of this app are not using the IPC sled then the IPC protocols should not be included with your code. You should not include the IPC Linea Pro libraries nor should you include "com.datecs.linea.pro.msr" and "com.datecs.linea.pro.bar" in the plist. Please note this is a common cause of rejection by Apple.

Establish Bluetooth Connection to the iMZ Printer and Print Sample ZPL Label Using GCD (Grand Central Dispatch)

```
#import <ExternalAccessory/ExternalAccessory.h>
#import "MfiBtPrinterConnection.h"
- (void) sendZplOverBluetooth {
    @autoreleasepool {
        //Dispatch this task to the default queue

dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT
, 0), ^ {
            NSString *serialNumber = @"";

            //Find the Zebra Bluetooth Accessory
            EAAccessoryManager *sam = [EAAccessoryManager
sharedAccessoryManager];
            NSArray * connectedAccessories = [sam connectedAccessories];

            for (EAAccessory *accessory in connectedAccessories) {
                if ([accessory.protocolStrings
indexOfObject:@"com.zebra.rawport"] != NSNotFound){
                    serialNumber = accessory.serialNumber;
                    break;
                    //Note: This will find the first printer connected!
If you have multiple Zebra printers connected, you should display a list
to the user and have him select the one they wish to use
                }
            }

            // Instantiate connection to Zebra Bluetooth accessory
            id<ZebraPrinterConnection, NSObject> thePrinterConn =
            [[[MfiBtPrinterConnection alloc]
initWithSerialNumber:serialNumber] autorelease];

            // Open the connection - physical connection is established
here.
            BOOL success = [thePrinterConn open];

            // This example prints "This is a ZPL test." near the top of
the label.
            NSString *zplData = @"^XA^FO20,20^A0N,25,25^FDThis is a ZPL
test.^FS^XZ";
            NSError *error = nil;

            // Send the data to printer as a byte array.
```

```
            success = success && [thePrinterConn write:[zplData
dataUsingEncoding:NSUTF8StringEncoding] error:&error];

            dispatch_async(dispatch_get_main_queue(), ^{
                if(success != YES || error != nil) {

                    UIAlertView *errorAlert = [[[UIAlertView alloc]
initWithTitle:@"Error" message:[error localizedDescription] delegate:nil
cancelButtonTitle:@"Ok" otherButtonTitles:nil] autorelease];

                    [errorAlert show];
                }
            });

            // Close the connection to release resources.
            [thePrinterConn close];
        });
    }
}
```

Establish Bluetooth connection to the iMZ printer and print sample CPCL label using GCD (Grand Central Dispatch)

```
#import <ExternalAccessory/ExternalAccessory.h>
#import "MfiBtPrinterConnection.h"
- (void) sendCpclOverBluetooth {
    @autoreleasepool {

dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT
, 0), ^ {
            NSString *serialNumber = @"";

            //Find the Zebra Bluetooth Accessory
            EAAccessoryManager *sam = [EAAccessoryManager
sharedAccessoryManager];
            NSArray * connectedAccessories = [sam connectedAccessories];

            for (EAAccessory *accessory in connectedAccessories) {
                if([accessory.protocolStrings
indexOfObject:@"com.zebra.rawport"] != NSNotFound){
                    serialNumber = accessory.serialNumber;
                    break;
                    //Note: This will find the first printer connected!
If you have multiple Zebra printers connected, you should display a list
to the user and have him select the one they wish to use
                }
            }

            // Instantiate connection to Zebra Bluetooth accessory
            id<ZebraPrinterConnection, NSObject> thePrinterConn =
[[[MfiBtPrinterConnection alloc] initWithSerialNumber:serialNumber]
autorelease];

            // Open the connection - physical connection is established
here.
            BOOL success = [thePrinterConn open];
```

```
            // This example prints "This is a CPCL test." near the top
of the label.
            NSString *cpclData = @"! 0 200 200 210 1\r\nTEXT 4 0 30 40
This is a CPCL test.\r\nFORM\r\nPRINT\r\n";
            NSError *error = nil;

            NSData *data = [NSData dataWithBytes:[cpclData UTF8String]
length:[cpclData length]];

            // Send the data to printer as a byte array.
            success = success && [thePrinterConn write:data
error:&error];

            dispatch_async(dispatch_get_main_queue(), ^{
                if(success != YES || error != nil) {
                    UIAlertView *errorAlert = [[[UIAlertView alloc]
initWithTitle:@"Error" message:[error localizedDescription] delegate:nil
cancelButtonTitle:@"Ok" otherButtonTitles:nil] autorelease];

                    [errorAlert show] ;
                }
            });

            // Close the connection to release resources.
            [thePrinterConn close];
        });
    }
}
```

Establish IPC connection to the iMZ printer using GCD (Grand Central Dispatch)

```
-(void)sampleWithGCD {
    @autoreleasepool {
        //Dispatch this task to the default queue

dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT
, 0), ^ {
            // Instantiate connection to Zebra Bluetooth printer through
a Linea IPC Sled
            id<ZebraPrinterConnection, NSObject> thePrinterConn =
[[[LineaBtPrinterConnection alloc] initWithMacAddress:@"001122334455"
andWithPin:@"1234"] autorelease];

            // Open the connection - physical connection is established
here.
            BOOL success = [thePrinterConn open];

            // This example prints "This is a ZPL test." near the top of
the label.
            NSString *zplData = @"^XA^FO20,20^A0N,25,25^FDThis is a ZPL
test.^FS^XZ";
            NSError *error = nil;

            // Send the data to printer as a byte array.
            success = success && [thePrinterConn write:[zplData
dataUsingEncoding:NSUTF8StringEncoding] error:&error];
```

```
            //Dispath GUI work back on to the main queue!
            dispatch_async(dispatch_get_main_queue(), ^{
                if (success != YES || error != nil) {
                    UIAlertView *errorAlert = [[[UIAlertView alloc]
initWithTitle:@"Error" message:[error localizedDescription] delegate:nil
cancelButtonTitle:@"Ok" otherButtonTitles:nil] autorelease];
                    [errorAlert show];
                }
            });

            // Close the connection to release resources.
            [thePrinterConn close];
        });
    }
}
```

Establish IPC connection to the iMZ printer and print sample ZPL label

```
#import <ExternalAccessory/ExternalAccessory.h>
#import "LineaBtPrinterConnection.h"
-(void)sendZplOverBluetooth{
    @autoreleasepool {
        // Instantiate connection to Zebra Bluetooth printer through a
Linea IPC Sled
        id<ZebraPrinterConnection, NSObject> thePrinterConn =
[[[LineaBtPrinterConnection alloc] initWithMacAddress:@"001122334455"
andWithPin:@"1234"] autorelease];

        // Open the connection - physical connection is established
here.
        BOOL success = [thePrinterConn open];

        // This example prints "This is a ZPL test." near the top of the
label.
        NSString *zplData = @"^XA^FO20,20^A0N,25,25^FDThis is a ZPL
test.^FS^XZ";
        NSError *error = nil;

        // Send the data to printer as a byte array.
        success = success && [thePrinterConn write:[zplData
dataUsingEncoding:NSUTF8StringEncoding] error:&error];
        if (success != YES || error != nil) {
            UIAlertView *errorAlert = [[[UIAlertView alloc]
initWithTitle:@"Error" message:[error localizedDescription] delegate:nil
cancelButtonTitle:@"Ok" otherButtonTitles:nil] autorelease];
            [errorAlert show];
        }
        // Close the connection to release resources.
        [thePrinterConn close];
    }
}
```

Establish IPC connection to the iMZ printer and print sample CPCL label

```
-(void)sendCpclOverBluetooth {
    @autoreleasepool{
        // Instantiate connection to Zebra Bluetooth printer through a
Linea IPC Sled
        id<ZebraPrinterConnection, NSObject> thePrinterConn =
[[[LineaBtPrinterConnection alloc] initWithMacAddress:@"001122334455"
andWithPin:@"1234"] autorelease];

        // Open the connection - physical connection is established
here.
        BOOL success = [thePrinterConn open];

        // This example prints "This is a CPCL test." near the top of
the label.
        NSString *cpclData = @"! 0 200 200 210 1\r\nTEXT 4 0 30 40 This
is a CPCL test.\r\nFORM\r\nPRINT\r\n";
        NSError *error = nil;

        // Send the data to printer as a byte array.
        success = success && [thePrinterConn write:[cpclData
dataUsingEncoding:NSUTF8StringEncoding] error:&error];
        if (success != YES || error != nil) {
            UIAlertView *errorAlert = [[[UIAlertView alloc]
initWithTitle:@"Error" message:[error localizedDescription] delegate:nil
cancelButtonTitle:@"Ok" otherButtonTitles:nil] autorelease];
            [errorAlert show];
        }

        // Close the connection to release resources.
        [thePrinterConn close];
    }
}
```

See Also

- For any further information, sample code and solutions or to request further content, visit the Zebra Developer Portal.
- For more information on ZPL/CPCL commands, download manuals from the Zebra website.

Document Control

Version	Date	Description
1	March, 2014	Initial release
2	2nd October 2017	Updated URLs

Disclaimer

All links and information provided within this document are correct at time of writing.

Created for Zebra Global ISV Program by Zebra Development Services.