Getting Started on iOS

Download Necessary Software

Your first step is to download and install the following:

- **Xcode** – make sure you download and install the latest update
  - **https://developer.apple.com/xcode/index.php**
- **AppLink™ 2.0 SDK** – download the Framework to an appropriate location on your file system.
  - [insert link to beta 2.0 download]

Configuring Xcode

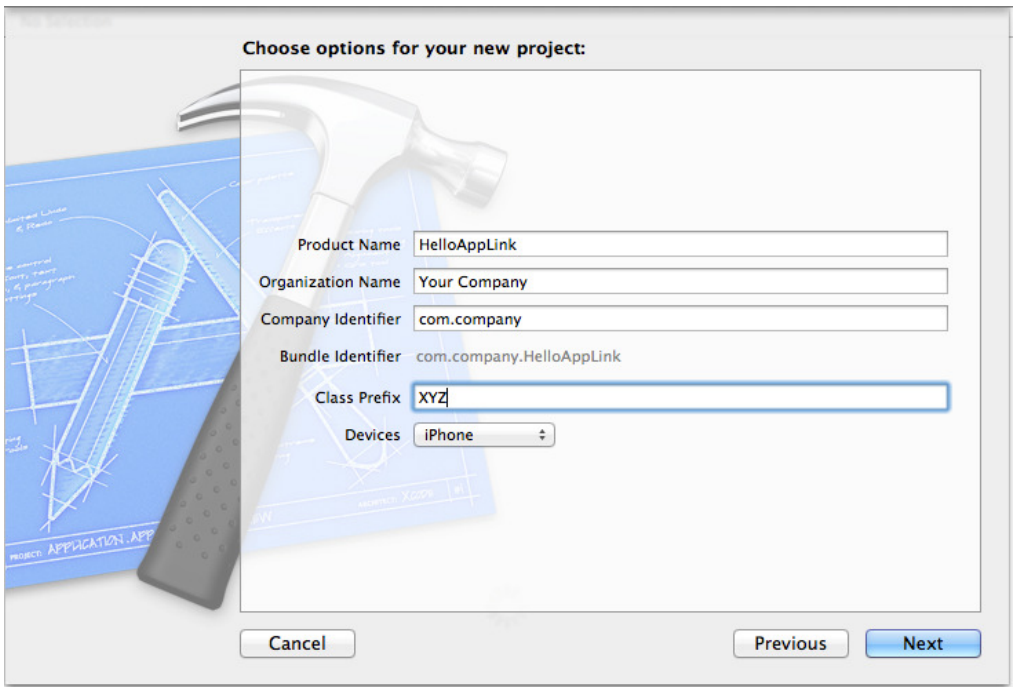
To run an application on an iOS device, you'll have to first set up your certificates, devices, provisioning profiles, etc.

For detailed information on doing this, visit the following Apple documentation: https://developer.apple.com/Library/ios/referencelibrary/GettingStarted/RoadMapiOS/chapters/RM_DevelopingForAppStore/DevelopingForAppStore/DevelopingForAppStore.html

Creating a New Project

If you already have a project, you can skip to the next section.

Open Xcode, and choose the **Single View Application** template. Enter your application's details. Change the device family to **iPhone**. Click **Next**, and then **Create**.

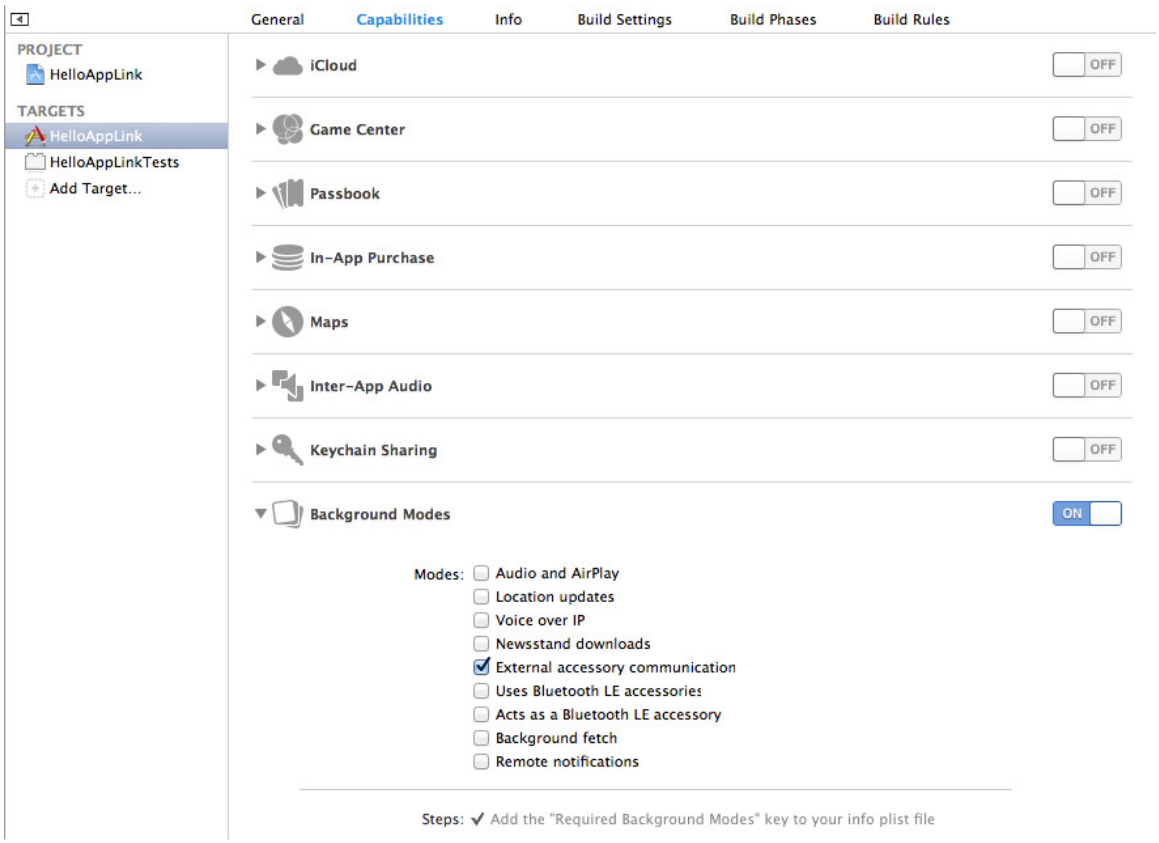


Enabling Background Capabilities

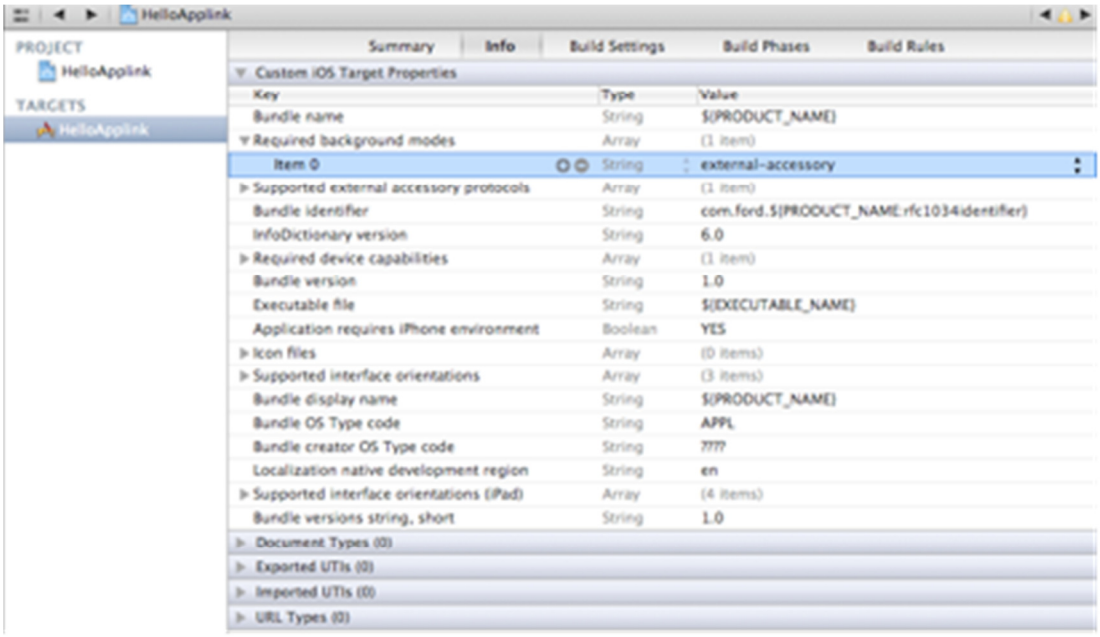
iOS 5 introduced the capability for an iOS application to maintain a connection to an external accessory while the application is in the background. This must be enabled manually and will only affect phones running iOS5+.

This can be enabled using one of two methods:

Method 1: Setting app Capabilities in build target settings
Click on your project in the Project Navigator, then select your app's build target. Click on the Capabilities tab and enable "Background Modes" and "External accessory communication":



Method 2: Directly modifying the .plist file
Navigate to your target's Info (or the .plist file) and add a row to the Custom iOS Target Properties section titled "Required background modes". Add an item to this array with the value "external-accessory":



Adding iOS 5+ Capabilities

Adding the AppLink Framework

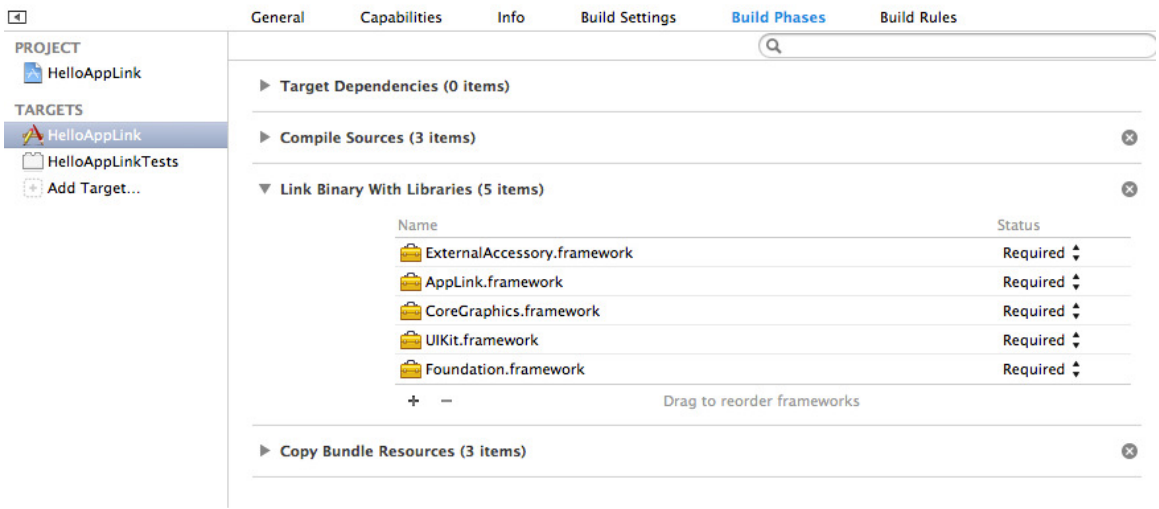
Navigate to your project folder in Finder. Copy the AppLink Framework to this location, or create a symbolic link from the framework location to your project folder.

Add the AppLink framework to the Linked Frameworks and Libraries section of the target by clicking the Add (+) button, then the “Add Other…” button and then select “Applink.framework”

Modifying Your Target's Settings

Build Phases

Add "ExternalAccessory.framework" under the "Linked Frameworks and Libraries" section:

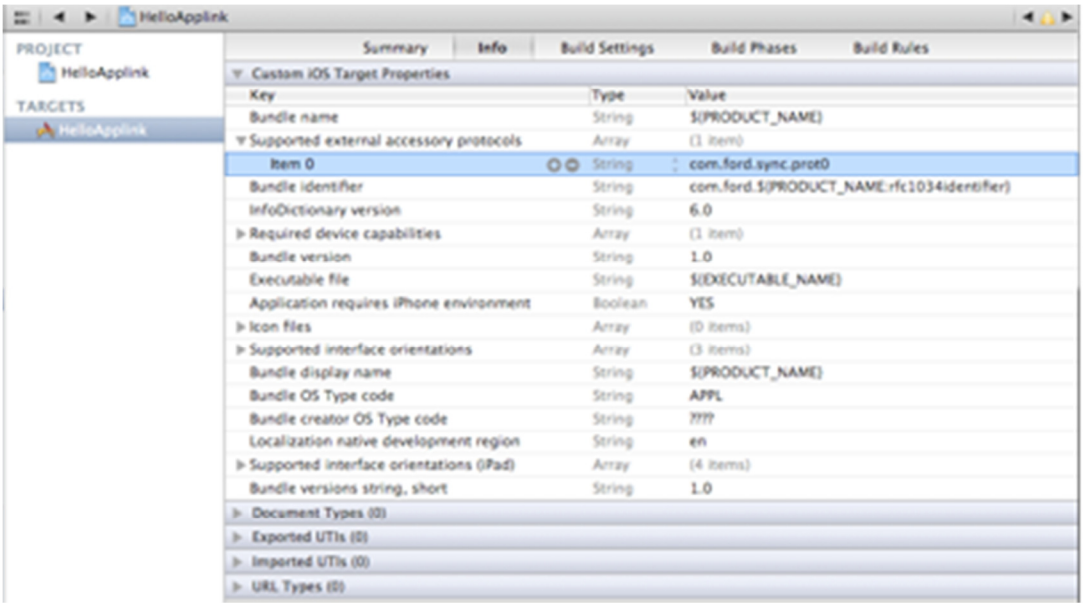


Targets Info

The SYNC® Protocol String is required in an AppLink™-enabled application's .plist file. Add a row titled "Supported external accessory protocols". To enable Multiple App Support (new with 2.3) add the following protocols:

```
com.ford.sync.prot0
com.smartdevicelink.prot0
com.smartdevicelink.prot1
com.smartdevicelink.prot2
com.smartdevicelink.prot3
com.smartdevicelink.prot4
com.smartdevicelink.prot5
com.smartdevicelink.prot6
com.smartdevicelink.prot7
com.smartdevicelink.prot8
com.smartdevicelink.prot9
com.smartdevicelink.prot10
com.smartdevicelink.prot11
com.smartdevicelink.prot12
com.smartdevicelink.prot13
com.smartdevicelink.prot14
com.smartdevicelink.prot15
com.smartdevicelink.prot16
com.smartdevicelink.prot17
com.smartdevicelink.prot18
com.smartdevicelink.prot19
com.smartdevicelink.prot20
com.smartdevicelink.prot21
com.smartdevicelink.prot22
com.smartdevicelink.prot23
com.smartdevicelink.prot24
```

```
com.smartdevicelink.prot25
com.smartdevicelink.prot26
com.smartdevicelink.prot27
com.smartdevicelink.prot28
com.smartdevicelink.prot29
```



Adding external accessory protocol

Adding AppLink™ Code

Ideally, all of the AppLink related code would be placed in its own class. However, for the sake of simplicity in sample code it may be located in a ViewController.

Bringing in FMCProxyListener

Add the following import where you will create the connection to AppLink:

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

Add the class variables to store the Proxy such as:

```
@property (strong, nonatomic) FMCSyncProxy* proxy;
@property (assign, nonatomic) int autoIncCorrID;
@property (assign, nonatomic) BOOL syncInitialized;
```

Add the stubs for the required FMCProxyListener's methods:

```
-(void) onOnHMIStatus:(FMOnHMIStatus*) notification {
}
```

```
-(void) onOnDriverDistraction:(FMOnDriverDistraction*)notification {
}
-(void) onProxyClosed {
}
-(void) onProxyOpened {
}
```

Creating Your SyncProxy

Next, you will create a function setupProxy():

```
-(void) setupProxy {
        self.proxy = [FMCSyncProxyFactory buildSyncProxyWithListener: self];
        self.autoIncCorrID = 101;
        [FMCDebugTool logInfo:@"Created SyncProxy instance"];
}
```

Implementing your onProxyOpened()

Implement the FMCProxyListener callbacks OnProxyOpened(), onError(), and onProxyClosed():

```
-(void) onProxyOpened {
        FMCRegisterAppInterface* regRequest = [FMCRPCRequestFactory
buildRegisterAppInterfaceWithAppName:@"Hello iPhone"
languageDesired:[FMCLanguage EN_US] appID:@"1234"];
        regRequest.isMediaApplication = [NSNumber numberWithBool:YES];
        regRequest.ngnMediaScreenAppName = nil;
        regRequest.vrSynonyms = nil;
        [consoleController appendMessage:regRequest];
        [self.proxy sendRPCRequest:regRequest];
}
-(void) onError:(NSException*) e {
        [FMCDebugTool logInfo:@"proxy error occurred: %@", e];
}
-(void) onProxyClosed {
        [FMCDebugTool logInfo:@"onProxyClosed"];
        [self tearDownProxy];
        [self setupProxy];
}
```

AppID is assigned to you through this developer site, and if you have not requested one yet, do so now through the following link:
[Insert Link]

Your apps will not work on Sync with out a valid AppName and AppID

Filling in your onOnHMIStatus()

By demonstrating a response to the onOnHMIStatus() message, we'll have shown that the application is correctly attached to AppLink and then you can begin to add real functionality.

```
-(void) onOnHMIStatus:(FMCOnHMIStatus*) notification {
	if (notification.hmiLevel == FMCHMILevel.HMI_NONE ) {
		// TODO:
	} else if (notification.hmiLevel == FMCHMILevel.HMI_FULL ) {
		[FMCDebugTool logInfo:@"HMI_FULL"];
		if (self.syncInitialized)
			return;
		self.syncInitialized = YES;
		FMCShow* msg = [FMCRPCRequestFactory
buildShowWithMainField1:@"Hi" mainField2:@"Mom!"
alignment:[FMCTextAlignment LEFT_ALIGNED] correlationID:[NSNumber
numberWithInt:self.autoIncCorrID++]];
		[self.proxy sendRPCRequest:msg];
	} else if (notification.hmiLevel == FMCHMILevel.HMI_BACKGROUND ) {
		// TODO:
	} else if (notification.hmiLevel == FMCHMILevel.HMI_LIMITED ) {
		// TODO:
	}
}
```

Making a tearDownProxy() Method

You'll also want a corresponding teardown function:

```
-(void) tearDownProxy {
	[FMCDebugTool logInfo:@"Disposing proxy"];
	[self.proxy dispose]; // required for both ARC and non-ARC
	//[self.proxy release]; // for non-ARC
	self.proxy = nil;
}
```

Setting up the Proxy

In your view controller or AppDelegate, create your proxy controller and call the setupProxy method:

```
- (void)viewDidLoad {
	[super viewDidLoad];
	if (!self.proxyController)
	{
```

```
            self.proxyController = [[XYZProxyController alloc] init];
        }
        [self.proxyController setupProxy];
}
```

Note: if you did not create a separate class for the proxy connection and put everything in a ViewController, the code above should instead contain:

```
- (void)viewDidLoad {
        [super viewDidLoad];
        [self setupProxy];
}
```

Building your Application

Make sure that your device is set to an iOS device and not simulator.

Launch the application through XCode by clicking the "Play" button.

Once the application is running on the device, you will want to launch the application on SYNC®. Follow the Launching an AppLink™ Application tutorial if you are unfamiliar with SYNC® HMI.

Don't forget to follow iOS developer guidelines

As an iOS developer you are also required to comply with all iOS developer program guidelines before submitting your application to Apple for review. There are iOS stipulations that you have to follow and it is up you as a developer to check the current Apple documentation and verify what guidelines are required. Understand that you are connecting to an external accessory over a USB connection and you need to follow iOS guidelines for connecting to an external device or accessory. Please review the iOS Developer Library to ensure you are meeting these requirements.