C# iOS and Android application for TSL 1128 handheld: Step by Step

# Abstract:

This document explains the step by step process of developing a mobile application for TSL 1128 handheld RFID/Barcode readers. It uses the C# language via the Xamarin framework to develop code that can run both on iOS and Android platforms.

# Installation and Import of the Reference Repo:

Follow the Installation instructions based on your development environment

* <https://developer.xamarin.com/guides/cross-platform/getting_started/requirements/>
* Follow instructions on the same website to setup your simulator and physical devices for mobile development.

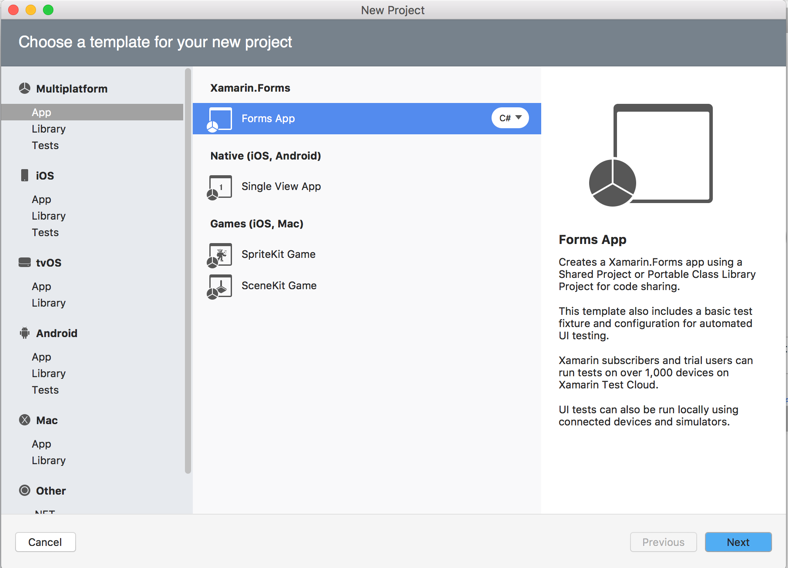
This document was developed with Xamarin Studio running on a Mac. On Windows, it is best to run Xamarin inside Visual Studio.

Clone the reference repo in Github.

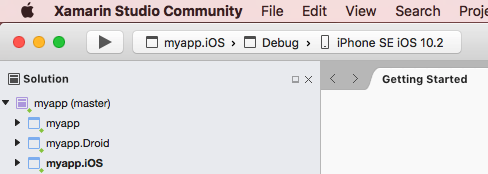
* <https://github.com/ralemy/xamarin-tsl-handheld>

# Setting Up your own project:

In your own repo (a separate directory), Create a new solution based on Xamarin Forms App:

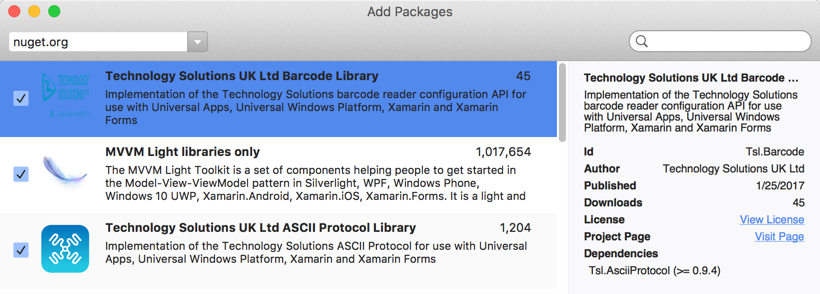


Give your app a name, I use “myapp” here, but it can be anything based on what you want your app to do.

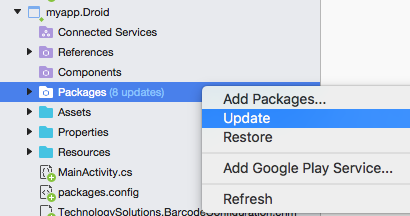


For all projects (myapp, myapp.Droid, myapp.iOS, etc) , right click and add the following packages:

* Tsl.Barcode, Tsl.AsciiProtocol, and MvvmLightLibs



Open each project and update the packages if necessary:

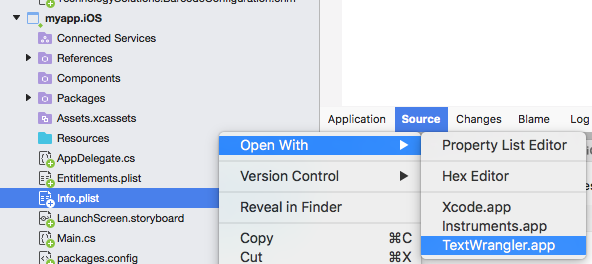


Add the following code to myapp.Droid /Properties/AndroidManifest.xml

<uses-permission android:name = "android.permission.BLUETOOTH" />

<uses-permission android:name = "android.permission.BLUETOOTH\_ADMIN" />

Edit myapp.iOS/info.plist with an external editor:



Add the following code to the dict element at the end of the file:

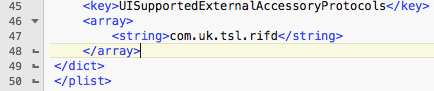
<key>UISupportedExternalAccessoryProtocols</key>

<array>

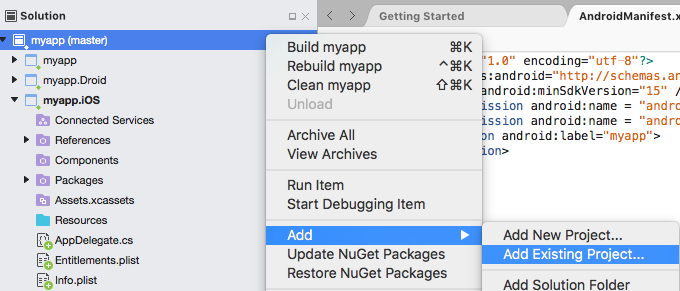
<string>com.uk.tsl.rifd</string>

</array>

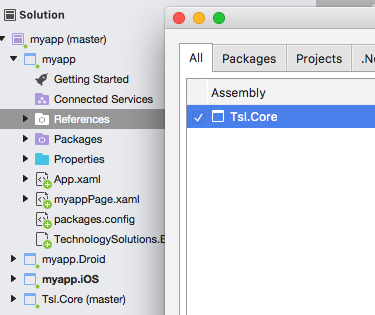
So your file will look like this:



# The next step is to import the Tsl.Core project into your solution. Right-click on the solution, select “add existing project”



Now navigate to the reference repo and select Tsl.Core/Tsl.Core.csproj. This will add Tsl.Core as the fourth component of your solution. Then go ahead and double click on references directory in your main project, and add Tsl.Core as a reference.



# How it is arranged:

This project uses the following technologies:

* Xamarin Forms, to create apps with C# that run both on iOS and Android.
* MVVMLight, for Dependency Injection and MVVM design pattern.
* TSL.AsciiProtocol and Tsl.Barcode for communication with the TSL reader.

When creating a new app, the steps explained in the “Setting Up Your Own Project” section will create a Xamarin Forms application which has both android and iOS sub projects. We will import the Tsl.Core project which provides us with general functionality for connecting to the handhelds and using the Mvvmlight framework. We then edit the main project to integrate that with the rest of our application.

# How it Works:

The main entry to the application is in Tsl/App.xaml.cs (shows as Tsl/App.xaml/App.xaml.cs in Xamarin Studio)

There are two things this file has to do, First, it has to make sure all needed dependencies are injected. Second, it has to register all required app pages.

For dependency injection, this project uses the SimpleIoc (inversion of control) class from the MvvmLight framework. Each plugin is expected to provide a class which implementes the Tsl.Core.IDependencyInjector Interface (you can find it in Tsl.Core/Interfaces directory)