STEP BY STEP GUIDE

FOR INTEGRATION AND USAGE OF
DIAGNOSTIC SDK/DLL

Document: Step by Step Guide for integration and usage of Diagnostic SDK/DLL
Version: 1.0
Author: Gurpreet Kaur
Designation: Senior Business Analyst

Contents

1. Introduction

Purpose

Welcome to the SDK documentation for the Xamarin Project SDK/DLL. This comprehensive document is aimed at providing step wise approach on consuming CellDe's SDK/DLL in Orange's existing Xamarin allocation/project. CellDe's SDK/DLL facilitates seamless integration of the powerful features into your application, enabling you to enhance user experiences and streamline development processes.

To fully leverage the capabilities of CellDe's diagnostic app within your existing application, add a reference to the app DLL in your Xamarin project. The diagnostic app has been developed using Visual Studio 2017, utilizing the cross-platform app feature with Xamarin.

Diagnostic App Overview

The device diagnostic app is an essential component of our SDK that allows diagnostics capabilities within your Xamarin projects. CellDe's mobile diagnostic app for smartphones and tablets is used to do an accurate assessment of the device's condition through a series of automated and interactive diagnostic tests and offer a better buyback value to the seller, thus reducing inspection cost for the business.

2. Environment Specifications

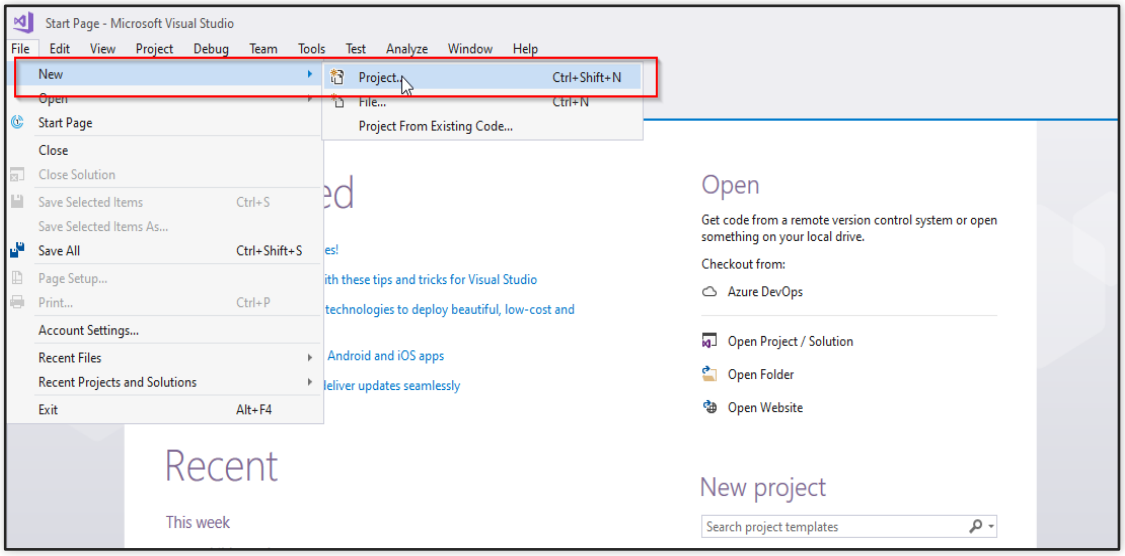
Operating System	Window 10/MAC
Visual Studio	2017
Xamarin	4.12.3.141
Xamarin.Android	9.1.7.0
Xamarin.iOS	12.4.0.64
Xamarin Templates	1.1.128
Xamarin Designer	4.16.30
Android: targetSdkVersion	27
XCODE	14

3. Step-by-Step Guide (CellDe Diagnostic App – POC)

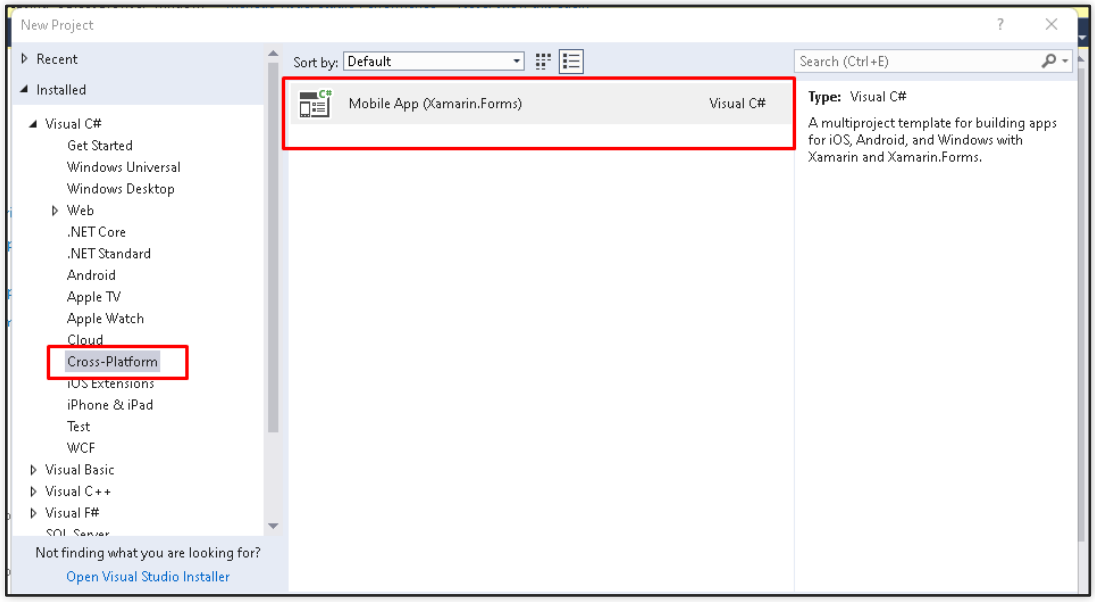
Creating the Diagnostic App

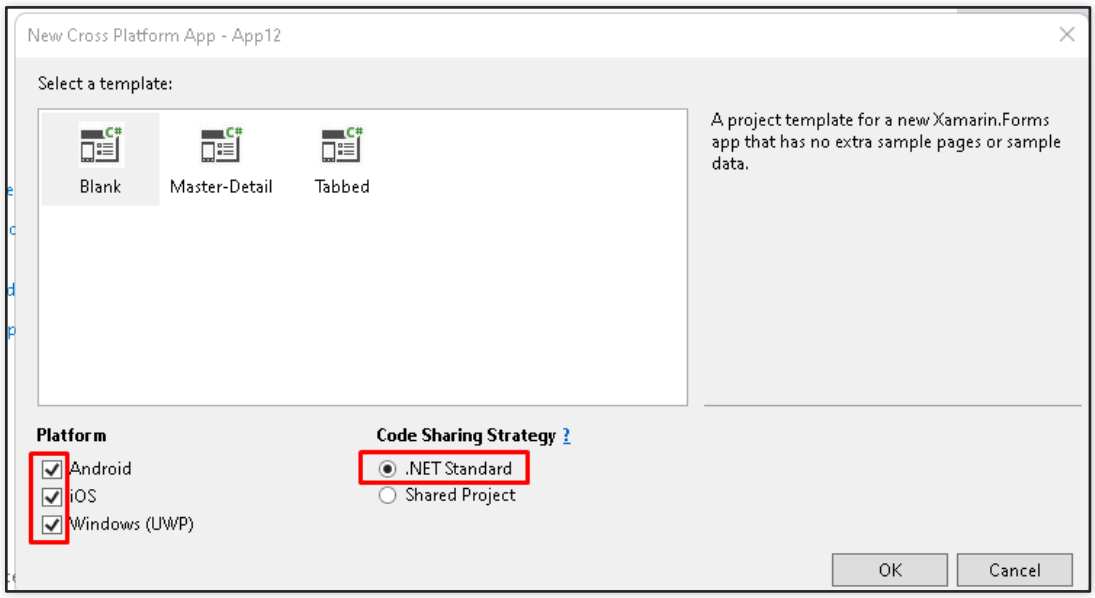
This section provides an overview of creating the diagnostic app and its key components.

1. **Creating the Mobile App Project:**
   Create a new project in Xamarin.Forms for your mobile app. This project will serve as the foundation for integrating the diagnostic app.

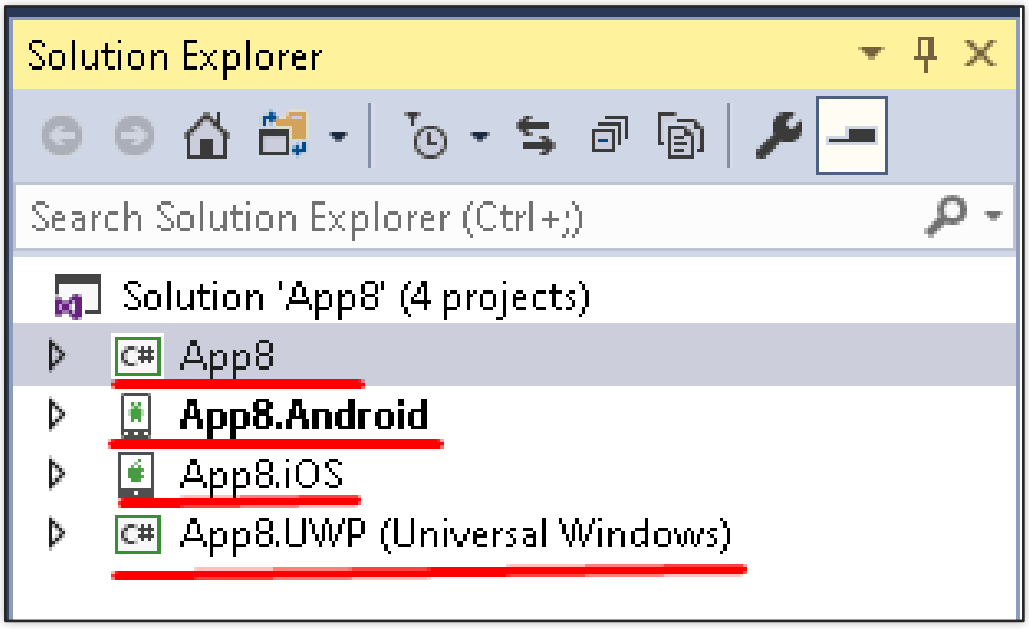


2. **Platform Selection:**
   Select the Cross-Platform Template. Choose the platform(s) on which you intend to develop your app. Xamarin.Forms supports multiple platforms including Android, iOS, and UWP (Universal Windows Platform).





3. **Solution Structure:**
   While examining the Solution Explorer, you will notice that your solution consists of multiple projects. In addition to your main project, for Android, there is an extra class library project called "App8" that contains code related to the user interface (UI) and its functionality. For iOS, the user interface and functionality is in its own project.



   a. **App8:** App8 is a class library that contains code related to the user interface (UI) and its functionality for Android devices. It serves as the core module for defining and implementing the UI elements and their behavior

b. **App8.Android:** This project is specific to Android devices. It focuses on implementing the backend code that complements the UI defined in the App8 class library. The project consumes the DLL generated from the App8 class library. In addition, specific permissions must be set in the AndroidManifest.xml file to accommodate the required inspection tests. For example, permissions related to Bluetooth functionality may be included:

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

c. **App8.iOS:** The App8.iOS project is dedicated to iOS devices. This contains the user interface and functionality for iOS device.

```
<key>NSBluetoothAlwaysUsageDescription</key>
    <string>Our app does not request this permission or utilize this
functionality but it is included in our info.plist since our app utilizes some
library, which references this permission in its code.</string>
    <key>NSBluetoothPeripheralUsageDescription</key>
<string>Our app does not request this permission or utilize this functionality
but it is included in our info.plist since our app utilizes some library,
which references this permission in its code.</string>
```

d. **App8.UWP (Optional):** App8.UWP is focused on supporting Windows devices that are compatible with the Universal Windows Platform. Code implementation for these devices is carried out in this project. (You can ignore this too since it will be used for Windows mobile)

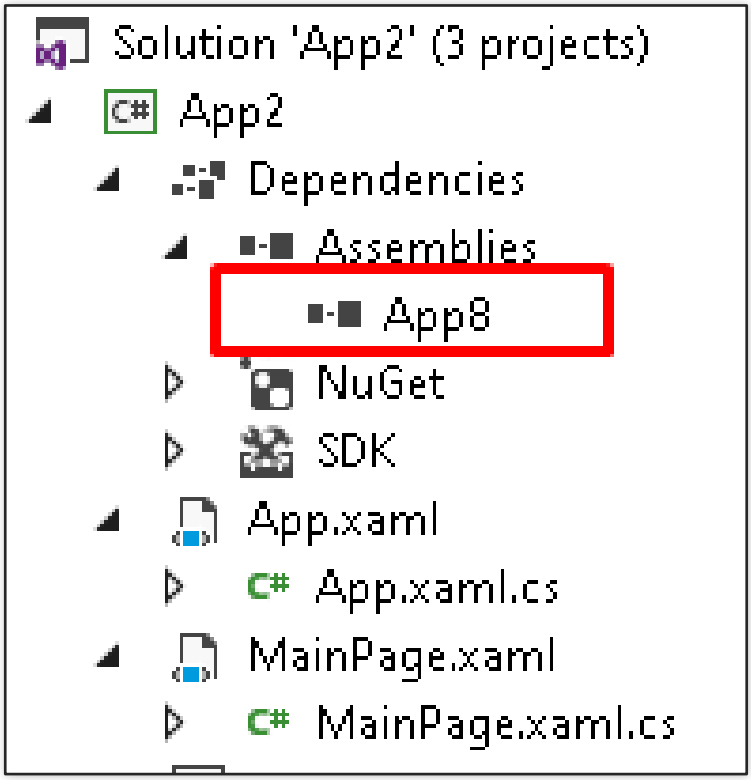
Getting Started (Consuming CellDe Diagnostic- POC)

To create a new Xamarin project that acts as a parent and consume the DLL of the child app, please refer the following steps:
Note: Follow the same steps for an existing Application

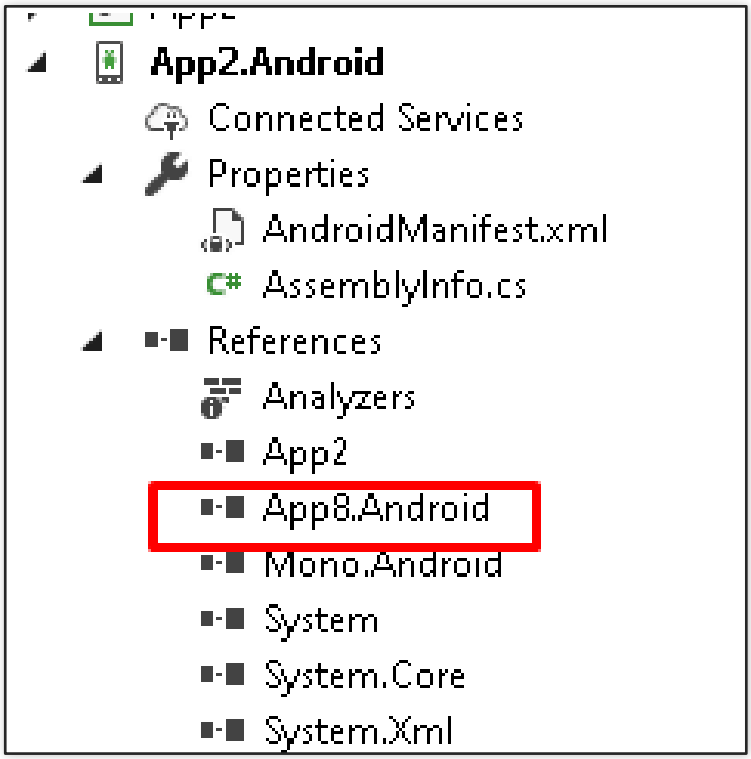
1. Create a new **Project** *(follow the same steps above)* or open an **existing Xamarin application** assuming the name of Project **(Parent App).** In case of existing Xamarin application we are assuming that project structure will be same as shown above.

Steps to consume Child App for Android

2. Add a reference to the class library (App8 **Child App**) in your new project **Parent App**. This will allow you to utilize the functionality provided by the App8 DLL within your project **(Parent App).**

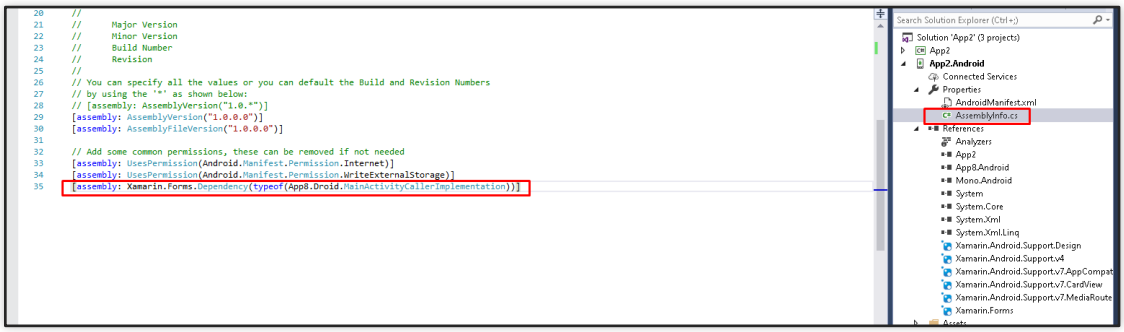


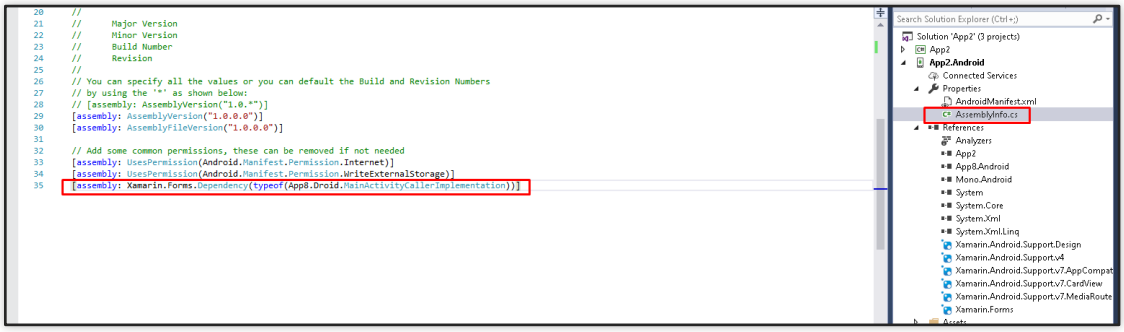
3. Add a reference to the Android project (App8.Android) in your new project class library (**Parent App**. Android). This step ensures that your parent project can access and utilize the backend code specific to Android devices that is defined in the App8 **ChildApp**.



4. In the AndroidManifest.xml file of your parent project, add necessary permissions required for the inspection tests to be performed. These permissions should align with the specific requirements of your diagnostic app. For example:

5. Add assembly inside AssemblyInfo.cs to make the dependency between parent and child Android app with the mentioned class “MainActivityCallerImplementation





Initialization and Usage Examples for Android

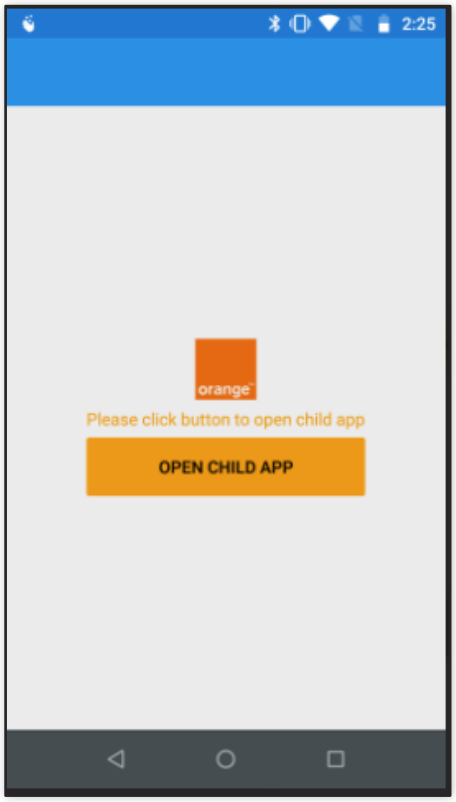
To initialize and configure the Child App (CellDe Diagnostic), please add the below line of code. Place this code on the page load event of your app or trigger it on a button click. Depending on your desired behavior,this code snippet will start the execution of the Child App and navigate to its initial page.

```
App8.MainPage childAppPage = new App8.MainPage();
await Navigation.PushAsync(childAppPage);
```

Note: Ensure to have the necessary references and dependencies properly set up in your Parent project to access the Child App's functionality.

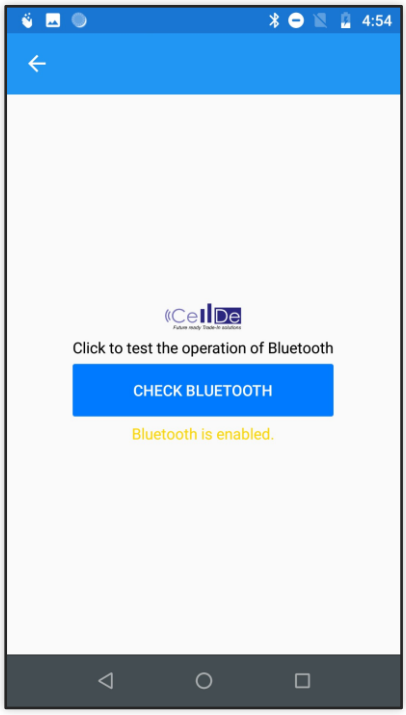
Below are the screenshots indicating the workflow of Parent App with its Child App.

Parent App

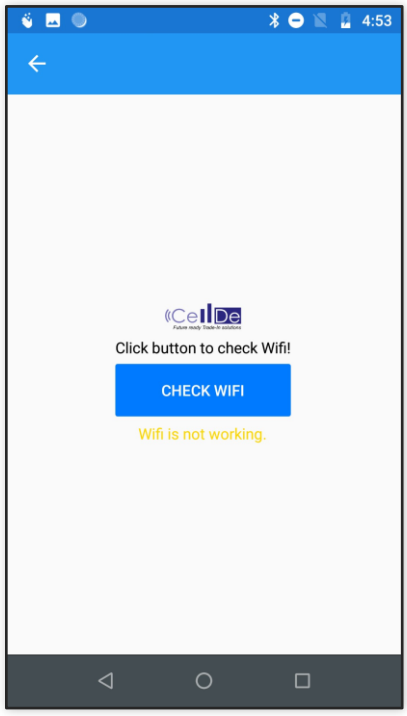


Child App

Bluetooth Test



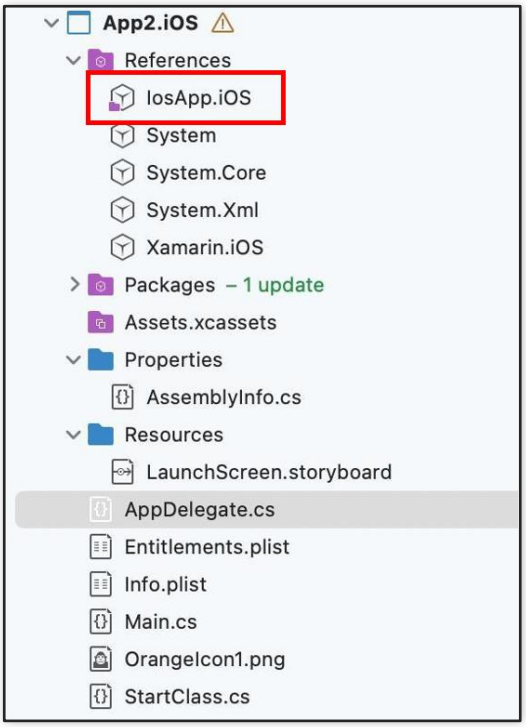
Wifi Test



Summary Screen

Steps to consume Child App for iOS

1. Add a reference to the iOS project (IosApp.iOS) in your new project class library (**Parent App**. iOS). This step ensures that your parent project can access and utilize functionality and user interface for iOS devices.



Initialization and Usage Examples for iOS

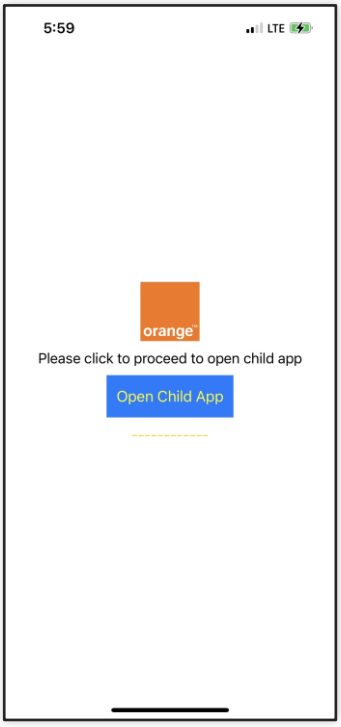
To initialize and configure the Child App (CellDe Diagnostic), please add the below line of code. Place this code on the page load event of your app or trigger it on a button click. Depending on your desired behavior, this code snippet will start the execution of the Child App and navigate to its initial page.

```
var contentPage = new YourContentViewController();
NavigationController.PushViewController(contentPage, true);
```

Note: Ensure to have the necessary references and dependencies properly set up in your Parent project to access the Child App's functionality.

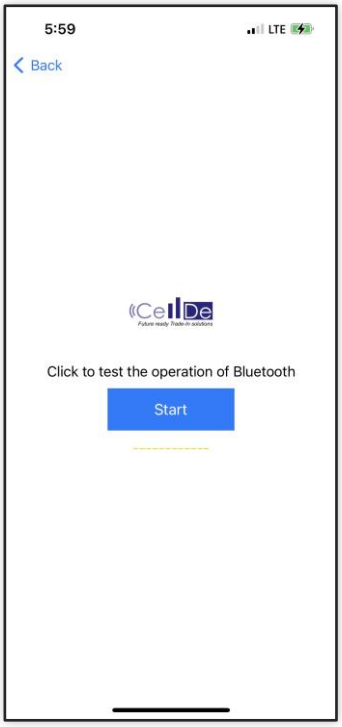
Below are the screenshots indicating the workflow of Parent App with its Child App.

Parent App



Child App

Bluetooth Test



Wifi Test

Summary Screen