Unity Bluetooth LE Plugin for iOS

Introduction

This plugin provides basic access to the CoreBluetooth API provided in iOS. You can use this plugin from Unity to create both Centrals and Peripherals. Some of the more advanced features of CoreBluetooth are not supported with this plugin.

For information about CoreBluetooth you will need to log into your Apple developer account and locate the CoreBluetooth documentation.

Version Changes

1.3

Added the characteristic UUID to the callback when receiving characteristic value updates

1.2

Fix a bug when passing null in the service UUIDs
Updated the documentation explaining adding CoreBluetooth Framework

Setup Guide

Setting up this plugin involves installing the package and writing scripts to interact with the plugin. Follow these steps:

1. Import the package into your Unity project
2. Create a a game object that you can attach a script to
3. Create a script that makes calls into the BluetoothHardwareInterface class using static methods for the Bluetooth functionality you require

Example Code

This package includes a sample scene with a simple GUI exposing the most common functions that you will need. The example is located under the Example folder.

The example scene just includes the Main Camera object that is created in a new project. The TestScript sample script has been attached to the main camera.

The TestScript builds a simple GUI. When each button is clicked some very basic code is executed to demonstrate using the basic features of this plugin.

The buttons on the left side of the screen are used to connect to Bluetooth LE peripherals that are near the device. This is known as being a Bluetooth LE Central.

The buttons on the right side of the screen are used to BE a Bluetooth LE Peripheral.

If the example project is built and deployed to 2 iOS devices the buttons on the right on one device can be used to become a peripheral and the buttons on the left of the second device can be used to connect to the first device.

All output showing the device states and connections and data transfer is sent to the debug output. In order to see this you must have an active debug session with Xcode or a console session using the Organizer tool in Xcode on a Mac.

Plugin Layout
The plugin has several parts.

Objective-C Code
The Plugins/iOS folder contains the Objective-C implementation. The full source code is contained in these files.

C# Scripts
The Plugins folder contains the C# scripts. These scripts provide the Unity – Plugin interface and some helper methods. The BluetoothHardwareInterface is a class that contains static methods to make calls into the Objective-C code. The BluetoothDeviceScript is used to receive messages passed back to Unity from the Obejctive-C code.

Example
The example code is detailed in the section above.

Editor Preprocessor
The Editor folder contains a preprocessor script that adds some required values to the plist.info file in the final Xcode project file.

CoreBluetooth Framework
In order to use Bluetooth Low Energy in an iOS app you must link your app with the CoreBluetooth framework library. This is done in Xcode and not in Unity. After you have built your app in Unity and opened the project in Xcode follow these steps to add the CoreBluetooth framework library:
1. Show the Project Navigator (upper left of tool panel the folder icon).
2. Select the Unity-iPhone Project at the top of the list. This will show the settings panel in the middle of the screen which has: General, Capabilities, Info, Build Settings, Build Phases and Build Rules.
3. Select the Build Phases
4. If it is not open already, open the Link Binary With Libraries section.
5. If CoreBluetooth is not in the list then click on the + symbol at the bottom of the list.

6. From the list that appears select the CoreBluetooth Framework and click on the Add button.

Every time you Replace the iOS project or build it from scratch you will need to go through these steps.

Support
For email support you can email support@shatalmic.com

API Reference

CoreBluetooth Enumeriations
public enum CBCharacteristicProperties
public enum CBAttributePermissions

BluetoothHardwareInterface Methods
public static void Log (string message)
       Log the string message to the Xcode console window

public static BluetoothDeviceScript Initialize (bool asCentral, bool asPeripheral, Action action, Action<string> errorAction)
       Initialize the Bluetooth system as either a central, peripheral or both.
       When completed the action callback will be executed. If there is an error the
       errorAction callback will be executed.

public static void DeInitialize (Action action)
       DeInitialize the Bluetooth system.
       When completed the action callback will be executed.

public static void FinishDeInitialize ()
       This method is automatically called by the BluetoothDeviceScript when it has
       been notified by the Objective-C code that everything else has been deinitialized.

public static void PauseMessages (bool isPaused)
       This method notifies the bluetooth system that the app is going to be paused or
       unpaused. While the app is paused incoming Bluetooth messages are cached in an
       array. When the app unpauses all cached Bluetooth messages are sent to the app.

public static void ScanForPeripheralsWithServices (string[] serviceUUIDs, Action<string, string> action)
       This method puts the device into a scan mode looking for any peripherals that
       support the service UUIDs in the serviceUUIDs parameter array. If serviceUUIDs
       is NULL all Bluetooth LE peripherals will be discovered. As devices are

discovered the action callback will be called with the ID and name of the peripheral.

public static void RetrieveListOfPeripheralsWithServices (string[] serviceUUIDs, Action<string, string> action)

     This method will retrieve a list of all currently connected peripherals with the UUIDs listed in the serviceUUIDs parameter. If serviceUUIDs is NULL all Bluetooth LE peripherals will be discovered. As devices are discovered the action callback will be called with the ID and name of the peripheral.

public static void StopScan ()

     This method stops the scanning mode initiated using the ScanForPeripheralsWithServices method call.

public static void ConnectToPeripheral (string name, Action<string> connectAction, Action<string, string> serviceAction, Action<string, string, string> characteristicAction)

     This method attempts to connect to the named peripheral. If the connection is successful the connectAction will be called with the name of the peripheral connected to. Once connected the serviceAction is called for each service the peripheral supports. Each service is enumerated and the characteristics supported by each service are indicated by calling the characteristicAction callback.

public static void DisconnectPeripheral (string name, Action<string> action)

     This method will disconnect a peripheral by name. When the disconnection is complete the action callback is called with the ID of the peripheral.

public static void ReadCharacteristic (string name, string service, string characteristic, Action<byte[]> action)

     This method will initiate a read of a characteristic using the name of the peripheral, the service and characteristic to be read. If the read is successful the action callback is called with the raw bytes of the read.

public static void WriteCharacteristic (string name, string service, string characteristic, byte[] data, int length, bool withResponse, Action<string> action)

     This method will initiate a write of a characteristic by the name of the peripheral and the service and characteristic to be written. The value to write is a byte buffer with the length indicated in the data and length parameters. The withResponse parameter indicates when the user wants a response after the write is completed. If a response is requested then the action callback is called with the message from the Bluetooth system on the result of the write operation.

public static void SubscribeCharacteristic (string name, string service, string characteristic, Action<string> notificationAction, Action<byte[]> action)

     This method will subscribe to a characteristic by peripheral name and the service and characteristic. The notificationAction callback is called when the notification

occurs and the action callback is called whenever the characteristic value is updated by the peripheral.

public static void UnSubscribeCharacteristic (string name, string service, string characteristic, Action<string> action)
This method is unsubscribe from a characteristic by name, service and characteristic. When complete the action callback is called.

public static void PeripheralName (string newName)
This method sets the peripheral name to use when advertising as a peripheral.

public static void CreateService (string uuid, bool primary, Action<string> action)
This method will create a service using the uuid parameter. It will be marked as primary or not by the primary parameter. When complete the action callback will be called. Note that the uuid can be a full UUID or a16 bit UUID. If the 16 bit UUID is used it will be converted into a full UUID according to the Bluetooth LE specification on using 16 bit UUID values.

public static void RemoveService (string uuid)
This method will remove a previously created peripheral service.

public static void RemoveServices ()
This method will remove all previously created peripheral services.

public static void CreateCharacteristic (string uuid, CBCharacteristicProperties properties, CBAttributePermissions permissions, byte[] data, int length)
This method will create a characteristic using the uuid parameter. It can have any of the CoreBluetooth properties and permissions contained in the enums. When complete the action callback will be called. Note that the uuid can be a full UUID or a16 bit UUID. If the 16 bit UUID is used it will be converted into a full UUID according to the Bluetooth LE specification on using 16 bit UUID values.

public static void RemoveCharacteristic (string uuid)
This method will remove a previously created peripheral characteristic.

public static void RemoveCharacteristics ()
This method will remove all previously created peripheral characteristic.

public static void StartAdvertising (Action action)
This method will start advertising as a Bluetooth LE peripheral using the services and characteristics created above.

public static void StopAdvertising (Action action)
This method will stop advertising as a Bluetooth LE peripheral.

public static void UpdateCharacteristicValue (string uuid, byte[] data, int length)

This method will update a characteristic's value. If a central has subscribed for notifications then it will receive a notification and value update.