RockMobile

The purpose of this document is to detail the APIs that has been built to aid in developing mobile apps for both iOS and Android devices.

Although multi-platform, naming conventions and design patters will more closely adhere to iOS / CSharp than they will Android.

Table of Contents

RockMobile 1

CoreLocation 1

iBeacon 1

## RockMobile

The primary problem when developing for multiple platforms is dealing with device specific features in a cross-platform way.

The goal of RockMobile is to provide unified, platform agnostic APIs for features that have different platform implementations.

For example, while both iOS and Android support iBeacon technology, they each have their own implementation with slightly different requirements. The iBeacon API offered by RockMobile abstracts the platform, allowing users to implement iBeacon support once and share the code across platforms.

## CoreLocation

Core Location represents the namespace for items that utilize location-detection technologies, such as GPS, iBeacon and Wifi.

### iBeacon

This namespace contains technology that allows platform agnostic use of iBeacons.

Usage:

Namespace: RockMobile.CoreLocation.iBeacon

Singleton: LocationManager.Instance

1. Create a new BeaconRegion containing the UUID of the iBeacons to monitor.

2. Add callbacks to LocationManager for AddRegionEnteredEvent and/or AddDidRangeBeaconsEvent

3. Call LocationManager’s StartMonitoring and StartRangingBeacons.

**Android Only:**

1. In the MainActivity, derive from IBeaconConsumer.
2. Cast LocationManager to DroidLocationManager and call BindIBeaconManager, passing the MainActivity’s ‘this’.
3. Implement IBeaconConsumer’s “public void OnIBeaconServiceConnect()”

and call DroidLocationManger.OnIBeaconServiceConnect

1. Implement OnResume() and call DroidLocationManager::EnterForegroundMode()
2. Implement OnStop() and call

DroidLocationManager::EnterBackgroundMode()

1. Implement OnDestroy() and call

DroidLocationManager::UnBindIBeaconManager()

For Background Scanning

1. Create a main Application class that derives from both Application and IBootstrapNotifier.
2. For Xamarin to consider your Application that main application,

override the standard constructor and call the base. Additionally, add the Application attribute.

1. Create a RegionBootstrap for any UUID that should be scanned.
2. In OnCreate(), allocate a region for the UUID that should be scanned, and pass it as an argument to a newly created RegionBootstrap object.
3. To control background scan time, get the IBeaconManager Instance in OnCreate, and set the background scan time and frequency.

Notes regarding background scanning:

Implemented via a bootstrapper service that is launched with your application. The user must reboot, plug in, or unplug their device to start the service.

Classes:

Beacon: An object storing the UUID, Major and Minor values.

BeaconRegion: Defines the UUID, and optionally Major, Minor values to “Range”.

RegionEventArgs: Contains the BeaconRegion of the region that was entered/exited.

RegionBeaconRangedEventArgs: Contains the BeaconRegion and associated Beacons that were “ranged”. (Meaning they were within the maximum distance to be discovered.)

LocationManager: The “core” object used for negotiating with the iBeaon technology.

Code Implementation:

iBeacon.cs – This implements the abstracted interface that end-users

should use.

iOS\_iBeacon.cs – This implements the iOS iBeacon API.

Apple’s CoreLocation framework contains CLLocationManager and

dependent classes. For the most part, iBeacon is a simple wrapper for

iOS\_iBeacon.

Droid\_iBeacon.cs – Because Android SDK doesn’t offer an “iBeacon” API, RockMobile uses RadiusNetwork’s SDK for Android. This mimics most of the functionality of the iOS implementation.

Because Android concepts such as ‘Binding’ objects, several additional features had to be added to the Android implementation. These are of course hidden from the iBeacon abstracted API, but include:

OnIBeaconServiceConnect() callback once Binding is finished

Queuing of monitor/ranging requests if binding has not completed.