# iOS FeasyBlue SDK API

## Reference Manual

## Version 1.0

# Revision History

| Version | Date | Notes | Author |
|---|---|---|---|
| 1.0 | 2018/8/1 | First Release | Liulian |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# Table of Contents

# 1. Introduction

This reference manual presents design guidelines for software engineers that use iOS FeasyBlue SDK to create iOS App for Bluetooth connectivity requirements.

## 1.1 iOS System Version Requirements

- iOS 8.0 and above

## 1.2 Supported iOS devices

- iPhone 5 and newer iPhone
- iPad mini and newer iPad mini
- iPad 3 and newer iPad
- iPod touch 6 and newer iPod touch

## 1.3 Supported Bluetooth Profile

- GATT (Generic Attribute Profile, relevant to BLE)
- iAP2 (iOS Accessory Protocol 2, relevant to MFi)

# 2. Get Started with FeasyBlue

## 2.1 General Tools

FeasyBeacon using the "pod" tool, and uses the MJRefresh, MBProgressHUD, SVProgressHUD and Masonry third-party tools, etc. Due to the use of the "pod" tool, when you run the project, please open the project with "xcworkspace "suffix.

## 2.2 FeasyBlue Demo App Project Setup

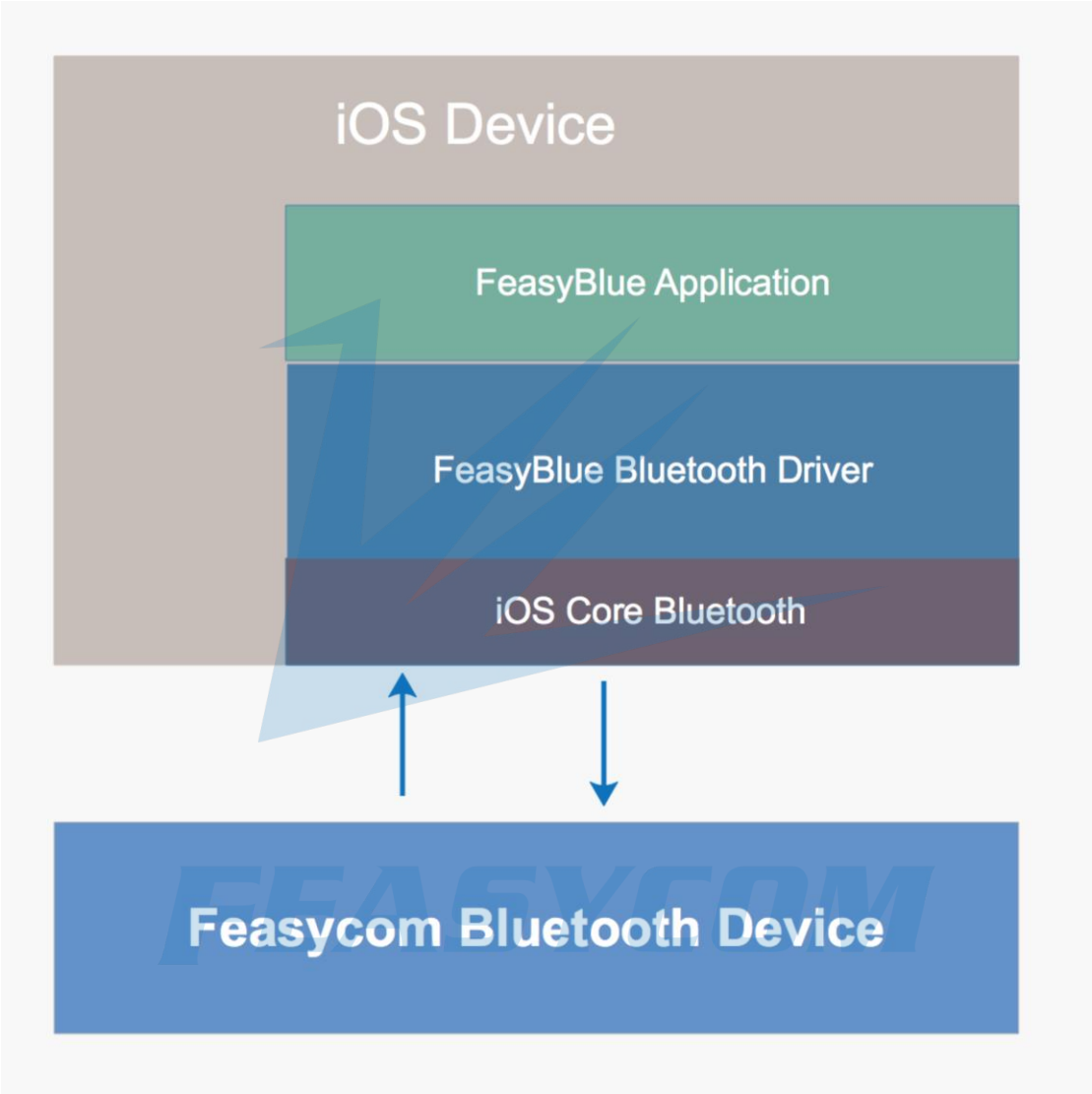
If you want to use the bluetooth function, add bluetooth permissions, TARGETS -> Info -> “Privacy - Bluetooth Peripheral Usage Description”, and If you want to bluetooth data transmission mode in the background, please open the background model, TARGETS -> Cacpbilities -> background modes -> Uses Bluetooth LE accessories.

## 2.3 Download and Run the FeasyBlue Demo App

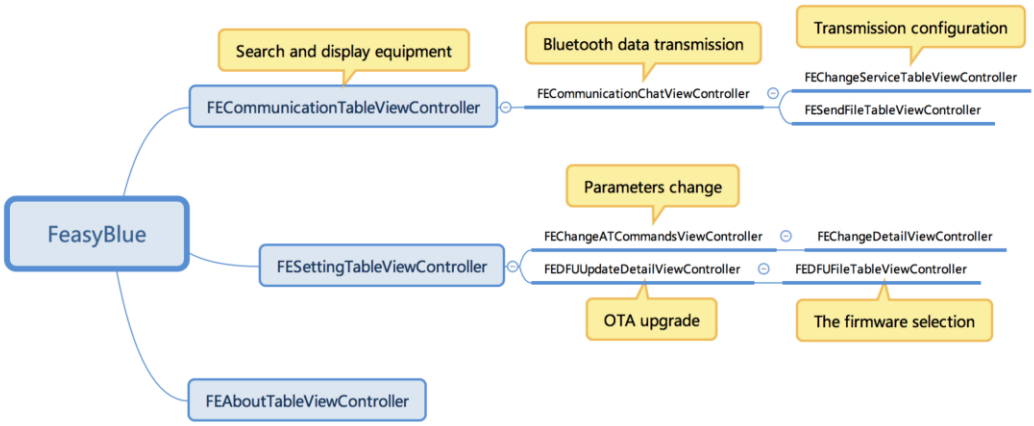
As a first test, we recommend you start with the Communication module. When the FeasyBlue App started, it runs the Communication module by default, and it will scan the nearby bluetooth devices automatically. Once there is a Feasycom bluetooth module displayed on the device scanning list, you can try to connect it if it is connectable. After FeasyBlue connected to a Feasycom bluetooth module, FeasyBlue will switch to a transmission page, then you can transferring data from or to bluetooth module.

# 3. FeasyBlue Architecture

## 3.1 Application Architecture

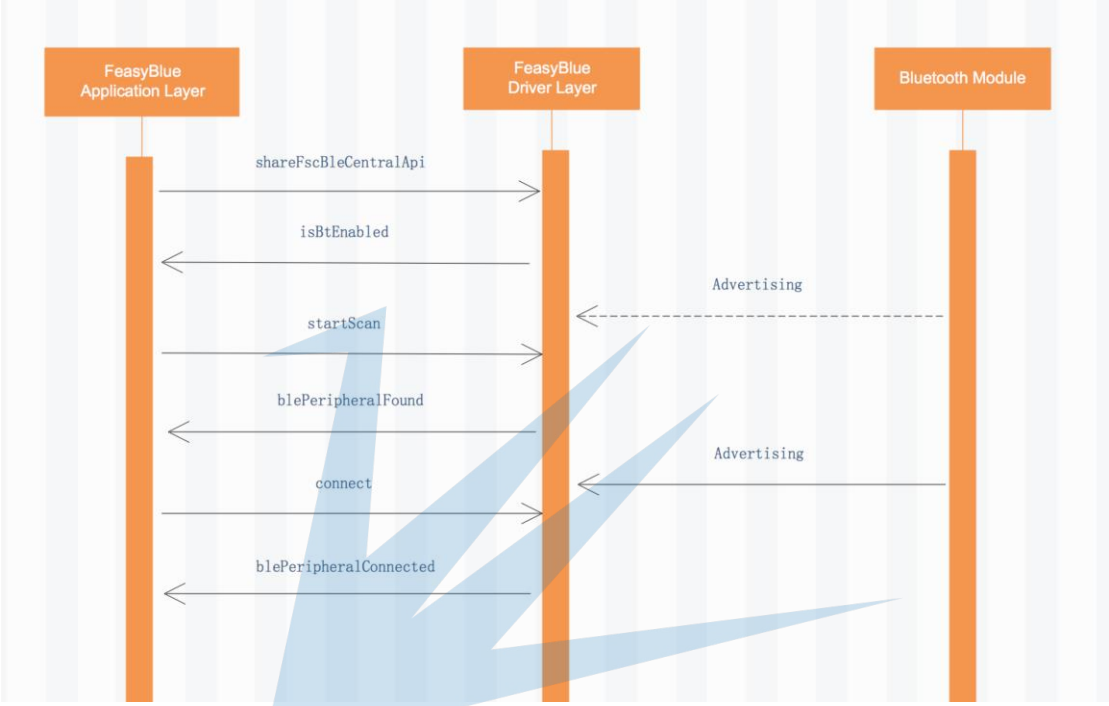


# 3.2 Page View Controller Topology



# 4. Operating Examples

## 4.1 Typical Initialization and Connection Setup



# 5. General APIs

## 5.1 ATTRIBUTES

```
/*
 * @property moduleType
 * @discussion                    Module type(BLE or Beacon).
 */
MODULETYPE moduleType
```

```
/*
 * @property peripheral
 * @discussion                    Last connected peripheral.
 */
CBPeripheral *peripheral
```

## 5.2 CALLBACKS

```
/*
 * @discussion                    Peripheral enabled callback, when the state of
 *                                central is CBManagerStatePoweredOn, call the
 *                                “startScan”  method.
 */
-(void)isBtEnabled:(void(^)(CBCentralManager *central))block
```

```
/*
 * Peripheral found callback,
 * @param central                 The central manager providing this update.
 * @param peripheral              A <code>CBPeripheral</code> object.
 * @param advertisementData       A dictionary containing any advertisement and scan
 *                                response data.
 * @param RSSI                    The current RSSI of <i>peripheral</i>, in dBm. A value of
 *                                <code>127</code> is reserved and indicates the RSSI was
 *                                not available.
 */
-(void)blePeripheralFound:(void(^)(CBCentralManager*central,CBPeripheral*peripheral,NS
Dictionary *advertisementData, NSNumber *RSSI))block
```

```
/*
 * Peripheral connected callback,
 * @param central                 The central manager providing this information.
 * @param peripheral              The <code> CBPeripheral </code> that has connected.
```

```
 * @discussion                  This method is invoked when a connection initiated by
 *                              {@link connect:} has succeeded.
 */
-(void)blePeripheralConnected:(void(^)(CBCentralManager*central,CBPeripheral*peripheral
))block
```

```
/*
 * Discover services callback,
 * @param services              The array of services information.
 * @param error                 If an error occurred, the cause of the failure.
 * @discussion                  This method returns the result of a @link
 *                              discoverServices @/link call. If the service(s) were read
 *                              successfully, they can be retrieved via.
 */
-(void)servicesFound:(void (^)(NSArray <CBService*>*services,NSError *error))block
```

```
/*
 * Peripheral disconnected callback,
 * @param central               The central manager providing this information.
 * @param peripheral            The <code>CBPeripheral</code> that has disconnected.
 * @param error                 If an error occurred, the cause of the failure.
 * @discussion                  This method is invoked upon the disconnection of a
 *                              peripheral that was connected by {@link
 *                              connect:}. If the disconnection was not
 *                              initiated by {@link disconnect}, the cause will be
 *                              detailed in the <i>error</i> parameter.  Once this
 *                              method has been.
 */
-(void)blePeripheralDisonnected:(void(^)(CBCentralManager*central,CBPeripheral*peripher
al, NSError *error))block
```

```
/*
 * Received packet callback,
 * @param peripheral            The peripheral providing this information.
 * @param characteristic        A <code>CBCharacteristic</code> object.
 * @param error                 If an error occurred, the cause of the failure.
 * @discussion                  This method is called when data is returned from the
 *                              peripheral.
 */
-(void)packetReceived:(void(^)(CBPeripheral*peripheral,CBCharacteristic*characteristic,NSE
rror*error))block
```

## 5.3 METHODS

| |
|---|
| /*<br>* @discussion The singleton. To initialize the<br>* <code>FscBleCentralApi</code>.<br>*/<br>+(instancetype)shareFscBleCentralApi |
| /*<br>* @discussion Start scan peripherals.<br>*/<br>-(void)startScan |
| /*<br>* @discussion Stop scan peripherals.<br>*/<br>-(void)stopScan |
| /*<br>* Connect peripheral,<br>* @param peripheral A <code> CBPeripheral </code> object.<br>* @discussion See “blePeripheralConnected:”.<br>*/<br>-(void)connect:(CBPeripheral *)peripheral |
| /*<br>* @discussion Disconnect peripheral.<br>*/<br>-(void)disconnect |

# 6. Communication APIs

## 6.1 METHODS

```
/* @param response              If yes, the <code>CBCharacteristicWriteWithResponse
 *                               </code> type is used, and if no, the
 *                               <code>CBCharacteristicWriteWithoutResponse</code>
 *                               type is used.
 * @param data                  The value to back.
 * @discussion                  This method is asynchronous, if you want to use
 *                               synchronized methods, see "syncSend: withResponse:".
 *                               Call this method before, please call the method
 *                               "setSendInterval:" once.
 */
-(void)send:(NSData*)data withResponse:(BOOL)response withSendStatusBlock:
(void(^)(NSData *data))block
```

```
/*
 * Send data to peripheral(sync),
 * @param data                  The value to write.
 * @param response              If yes, the <code>CBCharacteristicWriteWithResponse
 *                               </code> type is used, and if no, the
 *                               <code>CBCharacteristicWriteWithoutResponse</code>
 *                               type is used.
 * @discussion                  This method is synchronous, asynchronous method if you
 *                               want to use, see "send: withResponse:
 *                               withSendStatusBlock:".
 */
-(void)syncSend:(NSData *)data withResponse:(BOOL)response
```

```
/*
 * @discussion                  Stop send data to peripheral and reset sending status.
 */
-(void)stopSend
```

```
/*
 * Specify UUID to set characteristic,
 * @param serviceUUID           The UUID of service.
 * @param characteristicUUID    The UUID of characteristic.
 * @param notify                Whether listening to.
 * @param result                Whether to set up successfully.
 * @discussion                  This method allows you to specify UUID to search
 *                               services and characteristics.
 */
```

```
-(void)setCharacteristic:(NSString*)serviceUUID withCharacteristicUUID:(NSString
*)characteristicUUID withNotify:(BOOL)notify infoBlock:(void (^)(BOOL result))block
```

```
/*
  * Read characteristic value,
  * @param characteristic          A <code>CBCharacteristic</code> object.
  * @discussion                   Read the eigenvalue information manually, see the
  *                               method “readResponse:”.
  */
-(void)read:(CBCharacteristic *)characteristic
```

```
/*
  * Set send interval(ms),
  * @param interval               The gap between the packet.
  * @discussion                   If you want to call the method “send: withResponse:
  *                               withSendStatusBlock:”, please call this method once.
  */
-(void)setSendInterval:(NSInteger)interval
```

```
/*
  * Set mtu,
  * @discussion                   Call this method set data per packet size.
  */
-(void)setAttMtu:(NSInteger)mtu
```

## 6.2 CALLBACKS

```
/*
  * Peripheral disconnected callback,
  * @param characteristic          A <code>CBCharacteristic</code> object.
  * @param data                   The value to back.
  * @param error                  If an error occurred, the cause of the failure.
  * @discussion                   This method returns the result of a {@link send:
  *                               withResponse:} call, when the parameter “response”
  *                               is yes.
  */
-(void)sendCompleted:(void(^)(CBCharacteristic*characteristic,NSData*data,NSError*error))
block
```

```
/*
  * Response for characteristic value read,
  * @param characteristic          A <code>CBCharacteristic</code> object.
  * @discussion                   This method returns the result of a @link read: @/link
  *                               call.
  */
-(void)readResponse:(void(^)(CBCharacteristic*characteristic))block
```

# 7. Parameter Change APIs

## 7.1 METHODS

```
/*
 * Send AT commands,
 * @param commandArray          An array containing the AT commands.
 * @discussion                  See the callback method “fscAtResponse:”.
 */
-(void)sendFscAtCommands:(NSArray*)commandArray
```

## 7.2 CALLBACKS

```
/*
 * Response for send AT commands,
 * @param type                  A <code>CBCharacteristic</code> object.
 * @param status                status:OK; ERROR; TIMEOUT; ModifyNoNeed.
 * @discussion                  This method returns the result of a @link
 *                              sendFscAtCommands: @/link call.
 */
-(void)fscAtResponse:(void (^)(NSString*type,int status))block
```

# 8. Device Firmware Upgrade APIs

## 8.1 METHODS

```
/*
 * Load file and check file information,
 * @param dfuFileName          The name of the upgrade file.
 * @discussion                 Return a <code>NSDictionary</code> object. Include file
 *                             information.
 */
-(NSDictionary*)checkDfuFile:(NSString*)dfuFileName
```

```
/*
 * This method is called to upgrade,
 * @param dfuFileName          This parameter is the name of the upgrade file.
 * @param restore              Restore the factory settings.
 */
-(void)startOTA:(NSString*)dfuFileName withRestoreDefaultSettings:(BOOL)restore
```

## 8.2 CALLBACKS

```
/*
 * OTA update callbacks,
 * @param percentage           This parameter is the upgrade progress.
 * @param status               This parameter is the upgrade status.
 * @discussion                 This method returns the result of a @link startOTA:
 *                             withRestoreDefaultSettings: @/link call.
 */
-(void)otaProgressUpdate:(void (^)(CGFloat percentage, int status))block
```