Nut iOS SDK Document

Revision：

|  |  |  |
| --- | --- | --- |
| Time | Version | mark |
| 2016/12/29 | 1.1 | Init |
| 2020/11/19 | 1.2 | Support Beacon DFU |
| 2020/11/25 | 1.3 | Add DFU manual |

#### Development guide

**1. Development environment**

Xcode 4.0 or up; iOS OS 8.0 or up.

**2. Import SDK**

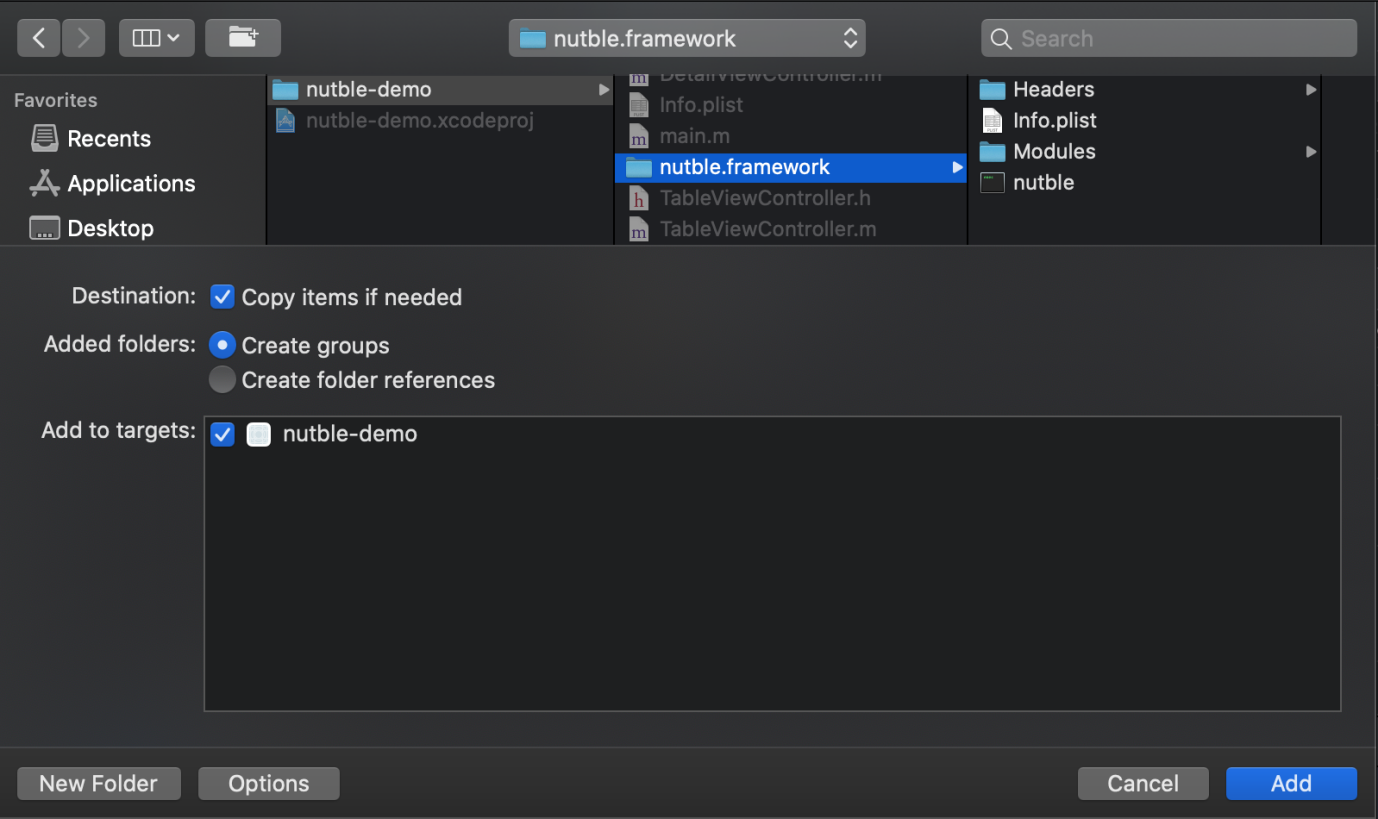
**2.1. Manually integrate dependent libraries**

2.1.1、Xcode File —> Add Files to "Your Project", Select the downloaded component package in the pop-up Panel -> Add. (Note: Check "Copy items if needed")

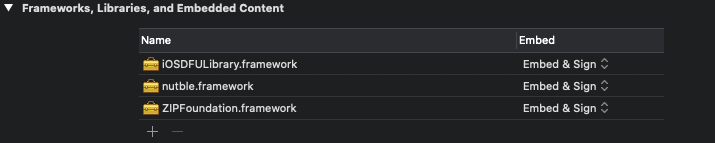
nutble.framework Nut SDK Core framework

iOSDFULibrary.framework Device DFU depends on the library

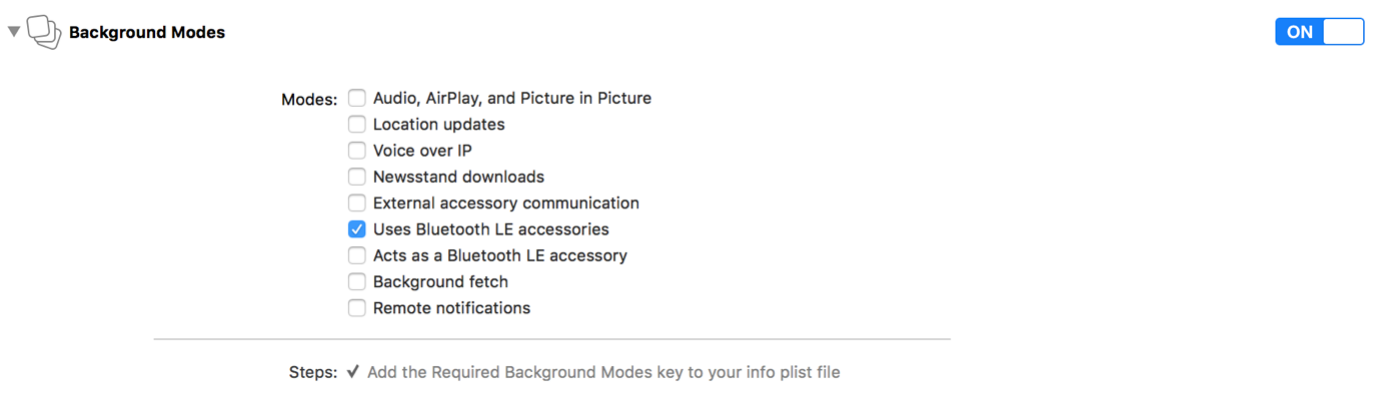
ZIPFoundation.framework The zip dependent library that the DFU lib depends on



2.1.2．Add dependent libraries, in the project settings target -> tab General ->Linked Frameworks and Libraries as follows:

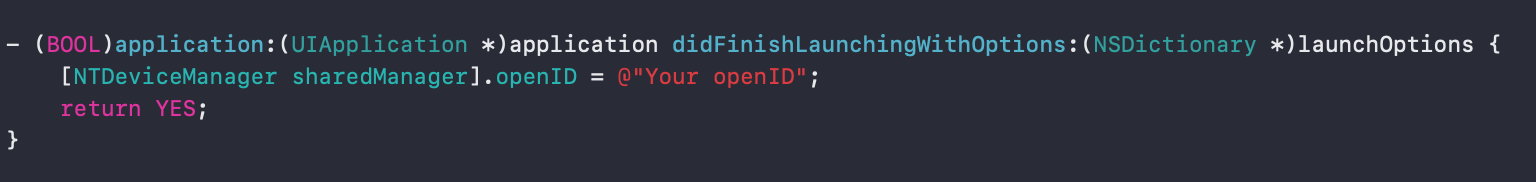


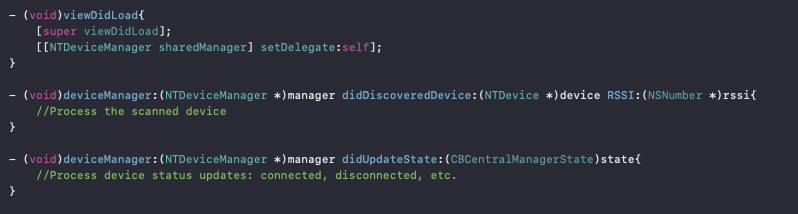
**2.2、Project configuration**

In the Background Modes in Capabiliies of the project configuration page, turn on "Use Bluetooth LE accessories". 

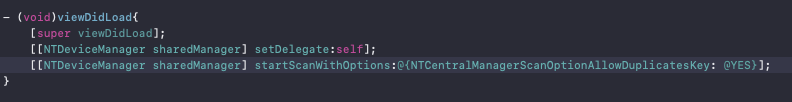
**3. Basic Feature**

3.1．Init the SDK

Initialize the NTDeviceManager object and set the openID, the following is the sample code:

3.2. Set up and implement NTDeviceManagerDelegate protocol

3.3. Enable to scan for nearby Nut Bluetooth devices



3.3.1．Scan API description

- (void)startScanWithOptions:(NSDictionary \*)optionDict

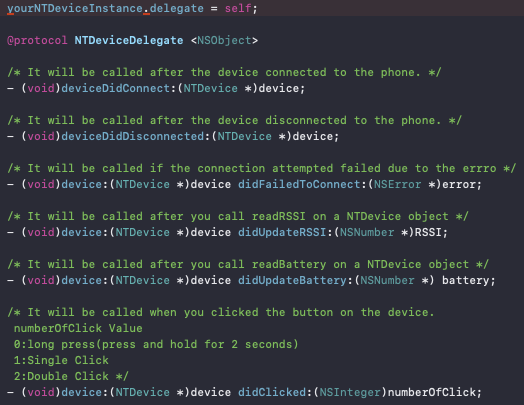
NTCentralManagerScanOptionAllowDuplicatesKey Whether to allow duplicate devices to be returned

If this option is YES, the device information will be returned through the interface every time a device is scanned.

If this option is NO, the broadcast of the device that is scanned for the first time will return the device information, and it will not be returned if it is scanned again.

**4. NTDevice related API description**

Set up and implement NTDeviceDelegate protocol



**- (void)connect ;**

Device connection: When the device is scanned, execute this method to connect to the device via Bluetooth.

If the device is successfully connected, the deviceDidConnect method will be called

The device is abnormally connected and the didFailedToConnect method will be called

**- (void)cancelConnection;**

Device disconnection: Perform this method to disconnect the device.

After the device is disconnected, the deviceDidDisconnected method will be called

**- (BOOL)shutdown;**

Device shutdown: Send a shutdown command to make the device enter the shutdown state. The device needs to be connected.

**- (BOOL)beep:(BOOL)enabled withTimeOutDuration:(NSInteger)timeout;**

Device find: Send a command to find device for phone to make the device play a sound alert. The device needs to be connected.

enable: YES Start find device; NO Stop find device.

timeout: Set the duration of find device for the phone, and it will stop automatically when the time is up.

**- (BOOL)setHardwareAlarmEnabled:(BOOL)enabled;**

Device alert: Send the device disconnect alert configuration, when the device is disconnected from the phone, the device will play alert. The device needs to be connected.

enable: YES turn on device disconnect alert; NO turn off device disconnect alert

Note: Disconnecting the device from the phone will distinguish between active and passive. Active disconnection of the phone from the device will not trigger this function. (The mobile phone is disconnected actively: App executes cancelConnection, phone turn off Bluetooth settings, etc.)

**- (void)readRSSI;**

Read the signal strength of the device, the signal strength value will be returned through the NTDeviceDelegate related method

**- (void)readBattery;**

Read the device power value, the power value will be returned through the NTDeviceDelegate related method

**- (BOOL)setBeaconUUID:(NSString \*)beaconUUID; (Only support customized devices)**

**- (BOOL)setBeaconMajor:(NSNumber \*)major minor:(NSNumber \*)minor;**

Set the UUID, Major, and Minor information of the device's Beacon until the effective Beacon information. After the device is restarted, if it is not connected within 30s, it will enter the Beacon working state.

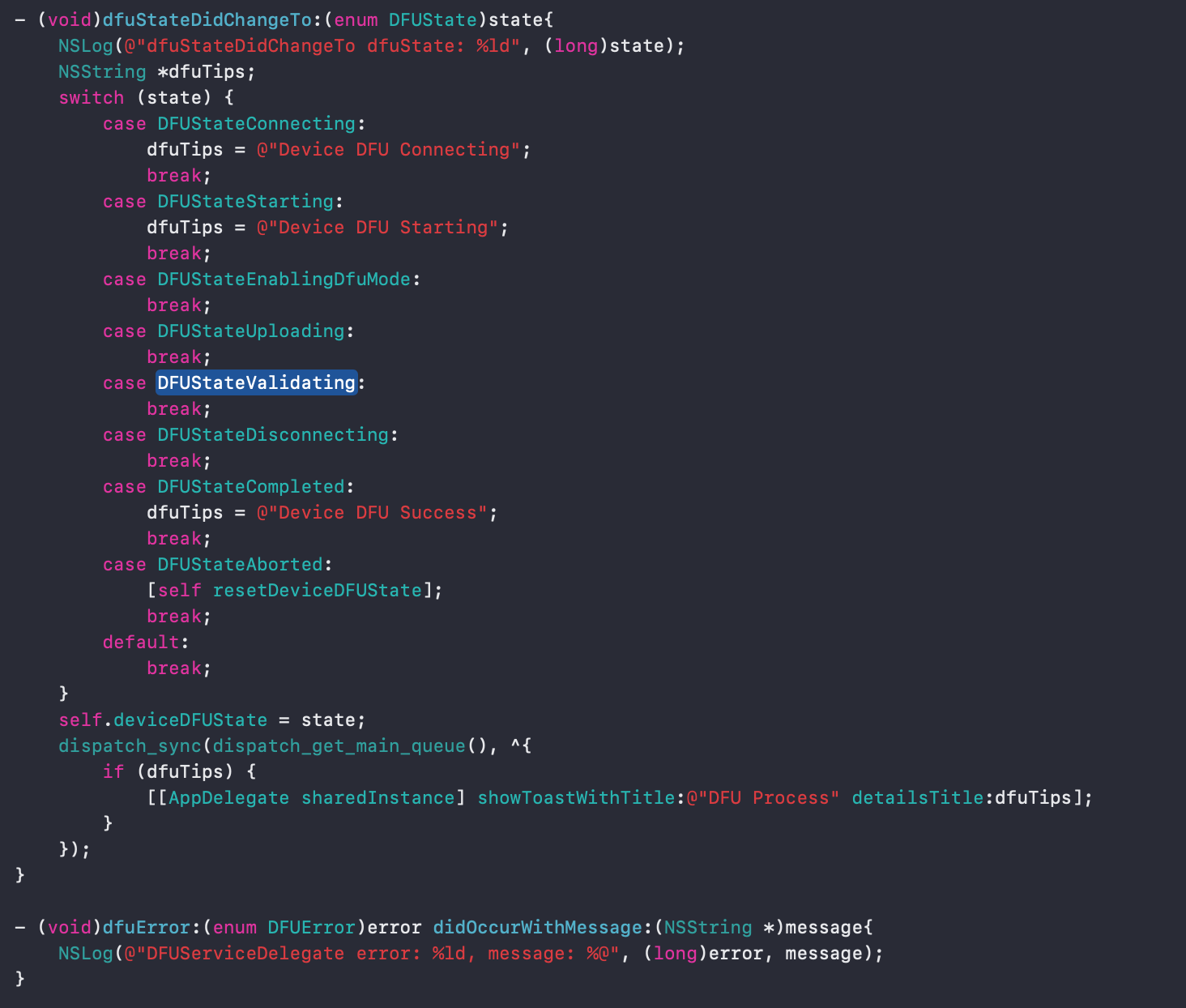
Set the UUID, Major, and Minor information of the device Beacon to FFF...FFF, and the device will enter the normal working mode after restarting.

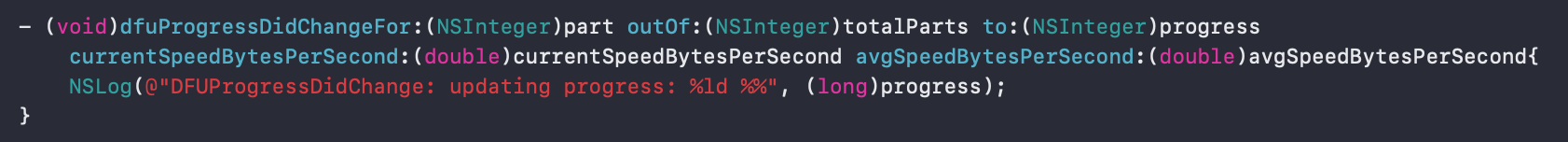
**- (BOOL)switchToDFUMode; (Only support customized devices)**

Set the device to enter DFU working mode: After executing this method, the device will actively disconnect from the mobile phone and enter DFU working mode.。

1. **DFU API description**

5.1. The DFUServiceDelegate and DFUProgressDelegate protocols are implemented. The DFUServiceDelegate protocol handles DFU status changes or error handling, and DFUProgressDelegate handles the DFU firmware sending process.





5.2. Initialize DFUFirmware, DFUServiceInitiator and DFUServiceController. This method is to initialize the firmware object through the firmware compression package, and then initialize the DFU service to start the upgrade process.



5.3 Introduction to equipment DFU process

5.3.1, turn on DFU mode

When the device enters the TAG working mode, connect the device and execute switchToDFUMode. The device will restart and then enter the DFU working mode. At this time, the Bluetooth broadcast name of the device is "DfuTarg", and the Mac address or DeviceId of the device will be incremented by 1.

5.3.2、Scan the device and start the upgrade

After discovering the device in DFU mode, call startDeviceDFU to start the DFU upgrade process. DFUServiceController will automatically connect to the device, send the firmware file, and the device will automatically restart after the sending is completed. At this time, the DFU process is completed.

### FAQ

1. How to obtain openID, please contact the SDK provider.

2. The use of the Beacon function, this function only supports customized devices, please contact the SDK provider to communicate specific requirements.

3. The use of DFU function, this function only supports customized devices, and has not yet supported Nut devices on sale.