

Directory

1. SDK Import instructions.....	1
2. The initialization.....	1
3. Connected devices.....	2
1. Setting Bluetooth Parameters.....	2
2. Set the Bluetooth discovery rules.....	2
3. Open the scanning.....	2
4. Connected devices.....	2
5. Close the scanning.....	3
4. Data sent.....	3
5. Data reception.....	3

1. SDK import instructions

1. Drag the PrinterSDK.framework from the zip package into your project

2. TARGETS -> Build Phases -> Link Binary With Libraries add libc++.tbd

3. TARGETS -> Build Settings -> Enable Bitcode Set to NO

4. TARGETS -> Build Settings -> Other Linker Flags add -ObjC

5. PCH file import #import <PrinterSDK/PrinterSDK.h>

2. The initialization

A singleton Bluetooth management class is recommended, BleManager is a property of the management class. See Demo for details

Initialization code

```
Manager.bleManager = [BleManager new];
```

3. Connected devices

1. Setting Bluetooth Parameters

Initialize the bleOptions class and set the bleOptions object to the Bluetooth management class. See Demo for details

```
BleOptions *options = [BleOptions new];{
    options.scanForPeripheralsWithOptions = @[CBCentralManagerScanOptionAllowDuplicatesKey:YES];
}
Manager.bleManager.bleOptions = options;
```

2. Set Bluetooth discovery rules

```
[Manager.bleManager setFilterOnDiscoverPeripherals:^(NSString * _Nonnull peripheralName, NSDictionary *
_Nonnull advertisementData, NSNumber * _Nonnull RSSI) {
    if (peripheralName.length) {
        return YES;
    }
}
```

```
        return NO;
    }
};
```

In general, the rule is set according to the BLE prefix. If not, all devices are found by default

3.Start scanning

Scanning on, the returned devices are returned according to the set rules

```
[Manager.bleManager scanPeripheralsWithBlock:^(CBCentralManager * _Nonnull central, CBPeripheral *
_Nonnull peripheral, NSDictionary * _Nonnull advertisementData, NSNumber * _Nonnull RSSI) {

}];
```

4.Connected devices

```
[Manager.bleManager connectToPeripheral:peripheral withConnected:^(CBCentralManager * _Nonnull central,
CBPeripheral * _Nonnull peripheral) {

} withFail:^(CBCentralManager * _Nonnull central, CBPeripheral * _Nonnull peripheral, NSError * _Nonnull error)

} withState:^(ConnectState state) {

}];
```

5.Close the scanning

```
[Manager.bleManager cancelScan];
```

4. Data sent

```
[Manager.bleManager write:data];

[Manager.bleManager write:data progress:^(NSUInteger total, NSUInteger progress) {

} receCallBack:^(NSData * _Nullable data) {

}];
```

5. Data reception

```
Manager.bleManager.readTotalCallback = ^(NSData * _Nullable data) {

};
```

```
[Manager.bleManager write:data progress:nil receCallBack:^(NSData * _Nullable data) {  
    }];
```