

Class

XBLEManager

The `XBLEManager` class handles Bluetooth communication and provides callback blocks for printer-related statuses and operations.

```
import Foundation
import CoreBluetooth
```

XBLEManagerDelegate

Delegate Methods

xbleDiscover(_:advertisementData:rssi:)

Discover Bluetooth Peripheral

```
func xbleDiscover(
    _ peripheral: CBPeripheral!,
    advertisementData: [AnyHashable : Any]!,
    rssi RSSI: NSNumber!
)
```

- **peripheral**
The discovered peripheral.
- **advertisementData**
The advertisement data of the peripheral.
- **RSSI**
The signal strength of the peripheral.

xbleConnect(_:)

Connection Successful

```
func xbleConnect(_ peripheral: CBPeripheral!)
```

- **peripheral**
The connected peripheral.

xbleFail(toConnect:error:)

Connection Failed

```
func xbleFail(  
    toConnect peripheral: CBPeripheral!,  
    error: Error!  
)
```

- **peripheral**
The peripheral that failed to connect.
- **error**
The error encountered during the connection attempt.

xbleDisconnectPeripheral(_:error:)

Disconnection

```
func xbleDisconnectPeripheral(  
    _ peripheral: CBPeripheral!,  
    error: Error!  
)
```

- **peripheral**
The disconnected peripheral.
- **error**
The error encountered during disconnection.

xbleWriteValue(for:error:)

Data Send Successful

```
func xbleWriteValue(  
    for characteristic: CBCharacteristic!,  
    error: Error!  
)
```

- **characteristic**
The characteristic used to write data.
- **error**
The error encountered during the write operation, if any.

xbleReceiveValue(for:error:)

Received Printer Data

```
func xbleReceiveValue(
    for characteristic: CBCharacteristic!,
    error: Error!
)
```

- **characteristic**
The characteristic from which data was received.
- **error**
The error encountered during the receive operation.

xbleCentralManagerDidUpdateState(_:)

Bluetooth Central Manager State Update

```
func xbleCentralManagerDidUpdateState(_ state: Int)
```

- **state**
The updated state of the central manager.

xbleFoundPeripheralServer(_:)

Found BLE Server

```
func xbleFoundPeripheralServer(_ peripheral: CBPeripheral!)
```

- **peripheral**
The found peripheral.

Properties

Bluetooth Communication Class

```
class XBLEManager: NSObject
```

- **name**
The name of the Bluetooth device.

```
var name: String?
```

- **isConnected**
The connection status of the Bluetooth device.

```
var isConnected: Bool = false
```

- **isScanning**

The current scanning status of the Bluetooth device.

```
var isScanning: Bool = false
```

- **writePeripheral**

The peripheral device associated with the write characteristic.

```
var writePeripheral: CBPeripheral?
```

- **write_characteristic**

The characteristic used to write data to the peripheral device.

```
var write_characteristic: CBCharacteristic?
```

- **read_characteristic**

The characteristic used to read data from the peripheral device.

```
var read_characteristic: CBCharacteristic?
```

- **notify_characteristic**

The characteristic used to receive notifications from the peripheral device.

```
var notify_characteristic: CBCharacteristic?
```

- **searchFilterUUID**

The UUID used to manually set the filter condition to find a specific printer device.

```
var searchFilterUUID: CBUUID?
```

- **characteristicUUID**

The UUID of the characteristic.

```
var characteristicUUID: CBUUID?
```

- **delegate**

The delegate that receives Bluetooth manager events.

```
weak var delegate: XBLEManagerDelegate?
```

Callback Blocks

- **receiveBlock**

The callback block called when data is received.

```
var receiveBlock: XBLEManagerReceiveCallbackBlock?
```

- **writeBlock**

The callback block called when data is written.

```
var writeBlock: XBLEManagerWriteCallbackBlock?
```

- **statusPOSBlock**

The callback block called when reporting POS printer status.

```
var statusPOSBlock: XBLEPOSPrinterStatusBlock?
```

- **statusLabelBlock**

The callback block called when reporting label printer status.

```
var statusLabelBlock: XBLELabelPrinterStatusBlock?
```

- **snBlock**

The callback block called when reporting the printer serial number.

```
var snBlock: XBLEPrinterSNBlock?
```

- **cashBoxBlock**

The callback block called when reporting cash box status.

```
var cashBoxBlock: XBLECashBoxBlock?
```

Sure, here's the provided Objective-C code converted to Markdown format:

Method

Singleton object

```
static let sharedInstance = XBLEManager()
```

Remove a delegate object

```
func removeDelegate(_ delegate: XBLEManagerDelegate!)
```

Remove all delegate objects

```
func removeAllDelegates()
```

Start scanning

```
func startScan()
```

Stop scanning

```
func stopScan()
```

Connect to a specific device

```
func connectDevice(_ peripheral: CBPeripheral!)
```

Manually disconnect the root peripheral

```
func disconnectRootPeripheral()
```

Send command (Write with Response)

```
func writeCommand(with data: Data!)
```

Send command with receive callback (Write with Response)

```
func writeCommand(  
    with data: Data!,  
    receiveCallBack receiveBlock: XBLEManagerReceiveCallBackBlock!  
)
```

Send command with write callback (Write with Response)

```
func writeCommand(  
    with data: Data!,  
    writeCallBack writeBlock: XBLEManagerWriteCallBackBlock!  
)
```

Send command (Write without Response)

```
func writeCommandDataWithoutResponse(_ data: Data!)
```

Set Bluetooth name and key

```
func setBluetoothNameAndKeyWith(  
    _ btName: String!,  
    btKey: String!  
)
```

Printer Status (for receipt printer)

```
func printerPOSStatus(_ statusBlock: XBLEPOSPrinterStatusBlock!)
```

Printer Status (for label printer)

```
func printerLabelStatus(_ statusBlock: XBLELabelPrinterStatusBlock!)
```

Printer Serial Number

```
func printerSN(_ snBlock: XBLEPrinterSNBlock!)
```

Cash Box Status

```
func cashBoxCheck(_ cashBoxBlock: XBLECashBoxBlock!)
```

Get Copyright Information

```
class func getCopyRight() -> String!
```