

Class

XUSBManager

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

```
#import <Foundation/Foundation.h>
#import <IOKit/usb/IOUSBLib.h>
```

XUSBManagerDelegate

Delegate Methods

xUSBConnected:

USB Connection Successful

```
- (void)xUSBConnected:(XUSBDeviceInfo *)device;
```

- **device**
The connected USB device information.

xUSBDisconnectWithError:message:device:

USB Disconnection with Error

```
- (void)xUSBDisconnectWithError:(int)code message:(NSString *)message device:
(XUSBDeviceInfo *)device;
```

- **code**
The error code indicating the reason for disconnection.
- **message**
The error message describing the disconnection.
- **device**
The disconnected USB device information.

xUSBWriteValueWithDevice:isSuccess:message:

USB Data Write Callback

```
- (void)xUSBWriteValueWithDevice:(XUSBDeviceInfo *)device isSuccess:(BOOL)isSuccess
message:(NSString *)message;
```

- **device**
The USB device information.

- **isSuccess**
Indicates whether the data write operation was successful.
- **message**
Additional message related to the write operation.

xUSBReceiveValueForData:device:

USB Data Receive Callback

```
- (void)xUSBReceiveValueForData:(NSData *)data device:(XUSBDeviceInfo *)device;
```

- **data**
The received data from the USB device.
- **device**
The USB device information.

xUSBDidSearch:device:

USB Device Search Callback

```
- (void)xUSBDidSearch:(int)state device:(XUSBDeviceInfo *)device;
```

- **state**
The search state: `0` for offline, `1` for online.
- **device**
The USB device information.

Properties

USB Communication Class

```
@interface XUSBManager : NSObject
```

- **searchBlock**
The callback block called when USB device search results are available.

```
@property (nonatomic, copy) UsbDidSearch searchBlock;
```

- **delegate**
The delegate that receives USB manager events.

```
@property (nonatomic, weak) id<XUSBManagerDelegate> delegate;
```

- **statusPOSBLOCK**
The callback block called when reporting POS printer status.

```
@property (nonatomic, copy) XUSBPOSPrinterStatusBlock statusPOSBlock;
```

- **statusLabelBlock**

The callback block called when reporting label printer status.

```
@property (nonatomic, copy) XUSBLabelPrinterStatusBlock statusLabelBlock;
```

- **snBlock**

The callback block called when reporting printer serial number.

```
@property (nonatomic, copy) XUSBPrinterSNBlock snBlock;
```

- **cashBoxBlock**

The callback block called when reporting cash box status.

```
@property (nonatomic, copy) XUSBCashBoxBlock cashBoxBlock;
```

Methods

Singleton Instance

```
+ (instancetype)sharedInstance;
```

- Returns the singleton instance of `XUSBManager`.

Remove a Delegate Object

```
- (void)removeDelegate:(id<XUSBManagerDelegate>)delegate;
```

- **delegate**

The delegate object to be removed.

Remove All Delegate Objects

```
- (void)removeAllDelegates;
```

Connect to a Specified USB Device

```
- (void)connectDevice:(XUSBDeviceInfo *)device;
```

- **device**

The USB device information to connect to.

Connect USB Device by Name

```
- (void)connectDeviceName:(NSString *)deviceName;
```

- **deviceName**

The name of the USB device to connect to.

Send Data

```
- (void)sendData:(NSData *)data;
```

- **data**

The data to be sent to the USB device.

Listen for USB Search Events

```
- (void)listenUsbSearchEvent:(UsbDidSearch)searchResult;
```

- **searchResult**

The callback block to handle USB device search results.

Disconnect a Specified USB Device

```
- (IOReturn)disconnect;
```

- Disconnects the currently connected USB device.

Search Devices

```
- (NSArray<XUSBDeviceInfo *> *)searchDevices;
```

- Returns an array of discovered USB devices.

Printer Status (for Receipt Printer)

```
- (void)printerPOSStatus:(XUSBPOSPrinterStatusBlock)statusBlock;
```

- **statusBlock**

The callback block to handle POS printer status updates.

Printer Status (for Label Printer)

```
- (void)printerLabelStatus:(XUSBLabelPrinterStatusBlock)statusBlock;
```

- **statusBlock**

The callback block to handle label printer status updates.

Printer Serial Number

```
- (void)printerSN:(XUSBPrinterSNBlock)snBlock;
```

- **snBlock**

The callback block to handle printer serial number updates.

Cash Box Status

```
- (void)cashBoxCheck:(XUSBCashBoxBlock)cashBoxBlock;
```

- **cashBoxBlock**

The callback block to handle cash box status updates.

XUSBDeviceInfo

Properties

- **vid**

The Vendor ID of the USB device.

```
@property (nonatomic, assign) uint16_t vid;
```

- **pid**

The Product ID of the USB device.

```
@property (nonatomic, assign) uint16_t pid;
```

- **locationId**

The Location ID of the USB device.

```
@property (nonatomic, strong) NSString *locationId;
```

- **deviceName**

The name of the USB device.

```
@property (nonatomic, strong) NSString *deviceName;
```

- **interface**

The USB interface for the device.

```
@property (nonatomic, assign) IOUSBInterfaceInterface245 **interface;
```

- **dev**

The USB device interface.

```
@property (nonatomic, assign) IOUSBDeviceInterface245 **dev;
```

- **pipeIn**

The input pipe for data transfer.

```
@property (nonatomic, assign) UInt8 pipeIn;
```

- **pipeOut**

The output pipe for data transfer.

```
@property (nonatomic, assign) UInt8 pipeOut;
```

USBConnectionError Enum

Error Codes

```
typedef NS_ENUM(NSInteger, USBConnectionError) {  
    USBConnectionErrorMasterPortCreationFailed = 1,           ///< Failed to create  
    master port.  
    USBConnectionErrorMatchingDictionaryCreationFailed,        ///< Failed to create  
    matching dictionary.  
    USBConnectionErrorVendorIDCreationFailed,                  ///< Failed to create  
    Vendor ID.  
    USBConnectionErrorProductIDCreationFailed,                 ///< Failed to create  
    Product ID.  
    USBConnectionErrorDevicePluginCreationFailed,              ///< Failed to create  
    device plugin.  
    USBConnectionErrorDeviceInterfaceCreationFailed,           ///< Failed to create  
    device interface.  
    USBConnectionErrorDeviceOpenFailed,                         ///< Failed to open  
    device.  
    USBConnectionErrorConfigurationCountFailed,                ///< Failed to get  
    configuration count.  
    USBConnectionErrorConfigDescriptorFailed,                  ///< Failed to get  
    configuration descriptor.  
}
```

```
        USBConnectionErrorSetConfigurationFailed,        ///< Failed to set
configuration.
        USBConnectionErrorInterfaceIteratorCreationFailed, ///< Failed to create
interface iterator.
        USBConnectionErrorInterfacePluginCreationFailed, ///< Failed to create
interface plugin.
        USBConnectionErrorInterfaceCreationFailed,        ///< Failed to create
interface.
        USBConnectionErrorInterfaceOpenFailed,            ///< Failed to open
interface.
        USBConnectionErrorEndpointCountFailed,            ///< Failed to get
endpoint count.
        USBConnectionErrorSetAlternateInterfaceFailed,    ///< Failed to set
alternate interface.
        USBConnectionErrorEndpointCountAltSettingFailed,  ///< Failed to get
endpoint count for alternate setting.
        USBConnectionErrorPipePropertiesFailed,            ///< Failed to get pipe
properties.
        USBConnectionErrorDeviceRemoved,                  ///< Device removed.
        USBConnectionErrorNormalDisconnection,            ///< Normal disconnection.
        USBConnectionErrorNotFoundDevice,                  ///< Device not found.
        USBConnectionErrorOtherDisconnection,              ///< Other disconnection
error.
};
```
