


Table of contents

第1章	Environment configuration instructions
第2章	SDK tools
第3章	SDK Bluetooth interface description
第4章	MicroLifeDevice interface description
第5章	Temperature interface description
第6章	Blood Pressure interface description
第7章	Weight interface description
第8章	Oxygen interface description
第9章	WatchBP O3 interface description
第10章	WatchBP Home interface description
第11章	WatchBP Office interface description
第12章	manual
第13章	Instructions for migrating V1.x to V2.x
第14章	Dome Code Description
第15章	Instructions
第16章	Update record

1. Environment configuration instructions:

- 1.1. Minimum supported version of SDK: **iOS 15**
- 1.2. Drag MicroLifeDeviceSDK.framework into the project.
- 1.3. Add MicroLifeDeviceSDK.framework in TARGETS/General/Frameworks,Libraries,and Embedded Content.
- 1.4. To use the body fat machine, you need to add BelterScaleInfo.framework in MicroLifeDeviceSDK to Frameworks, Libraries, and Embedded Content and set "Do Not Embed".

Frameworks, Libraries, and Embedded Content

Name	Embed
 BelterScaleInfo.framework	Do Not Embed ↕
 ECGProcess.framework	Do Not Embed ↕
 MessageUI.framework	Do Not Embed ↕
 MicroLifeDeviceSDK.framework	Do Not Embed ↕
 ZipArchive.framework	Embed & Sign ↕
+ -	

- 1.5. Add -all_load in TARGETS/Build Settings/Other Linkeder Flags.
- 1.6. Wherever you need to use it, just introduce the header file.

```
#import <MicroLifeDeviceSDK/MicroLifeDeviceSDK.h>
```

2. SDK tool description:

1.7. Log Management: Local Log Recording Tool

```
//Log Management
#import <MicroLifeDeviceSDK/LogManager.h>
```

1.8. Bluetooth Core :

- 1.8.1. BLESDK: Bluetooth underlying manager, responsible for all common Bluetooth operations.
- 1.8.2. BLEDeviceInfo: Bluetooth device tool, Bluetooth scan response basic objects.
- 1.8.3. MicroLifeDevice: Bailuo universal device tool. Responsible for the transmission and analysis of instructions for Bailuo's various equipment.
- 1.8.4. MicroLifeDataModel: Universal data object.
- 1.8.5. MicroLifeDeviceInfo: Bailuo device object.
- 1.8.6. MicroLifeUserInfo: Bailuo user object.

```
//Bluetooth Core
#import <MicroLifeDeviceSDK/BLESDK.h>
#import <MicroLifeDeviceSDK/BLEDeviceInfo.h>
#import <MicroLifeDeviceSDK/MicroLifeDevice.h>
#import <MicroLifeDeviceSDK/MicroLifeDataModel.h>

//Data Model
#import <MicroLifeDeviceSDK/MicroLifeDeviceInfo.h>
#import <MicroLifeDeviceSDK/MicroLifeUserInfo.h>
```

1.9. Temperature :

- 1.9.1. MicroLifeTemperature: MicroLifeTemperature device tool.
- 1.9.2. MicroLifeTemperatureMeasureData : Measure Data

```
//Temperature
#import <MicroLifeDeviceSDK/MicroLifeTemperature.h>
#import <MicroLifeDeviceSDK/MicroLifeTemperatureMeasureData.h>
```

1.10. Blood Pressure :

- 1.10.1. MicroLife3GBP: 3G BP device tool.
- 1.10.2. MicroLife4GBP: 4G BP device tool.
- 1.10.3. MicroLifeBloodPressureDRecord : DRecord。
- 1.10.4. MicroLifeBloodPressureCurrentAndMData : Current, MData。

```
//Blood Pressure
#import <MicroLifeDeviceSDK/MicroLife3GBP.h>
#import <MicroLifeDeviceSDK/MicroLife4GBP.h>
#import <MicroLifeDeviceSDK/MicroLifeBloodPressureDRecord.h>
#import <MicroLifeDeviceSDK/MicroLifeBloodPressureCurrentAndMData.h>
```

1.11. Weight :

- 1.11.1. MicroLifeWeight: Weight device tool.
- 1.11.2. MicroLifeBodyFat : Weight data。
- 1.11.3. BelterScaleInfo.framework: Body composition analysis tool.

```
//Weight
#import <MicroLifeDeviceSDK/MicroLifeWeight.h>
#import <MicroLifeDeviceSDK/MicroLifeBodyFat.h>
```

1.12. Oxygen :

- 1.12.1. MicroLifeOxygen: Oxygen device tool.
- 1.12.2. MicroLifeOxygenData : Oxygen data。

```
//Oxygen
#import <MicroLifeDeviceSDK/MicroLifeOxygen.h>
#import <MicroLifeDeviceSDK/MicroLifeOxygenData.h>
```

1.13. WatchBP :

- 1.13.1. MicroLifeWatchBPHome: WatchBP Home device tool.
- 1.13.2. MicroLifeWatchBP03: WatchBP O3 device tool.
- 1.13.3. MicroLifeWatchBPOffice: WatchBP Office device tool.
- 1.13.4. MicroLifeWatchBPDRecord : DRecord。
- 1.13.5. MicroLifeWatchBPCurrentAndMData : Current、MData。
- 1.13.6. MicroLifeWatchBPCBPdataAndCalCBP : CBP data、CalCBP。
- 1.13.7. MicroLifeWatchBPSettingValues : Setting Values。
- 1.13.8. MicroLifeWatchBPFunctionSettingValues : Function Setting Values。
- 1.13.9. MicroLifeWatchBPDDiagnosticDRecord : Diagnostic DRecord。
- 1.13.10. MicroLifeWatchBPNocturnalModeDRecord : Nocturnal Mode DRecord。
- 1.13.11. MicroLifeWatchBPRemoteMeasurement : Remote Measurement。
- 1.13.12. MicroLifeWatchBPM1: WatchBP M1 device tool.

```
//WatchBP
#import <MicroLifeDeviceSDK/MicroLifeWatchBPHome.h>
#import <MicroLifeDeviceSDK/MicroLifeWatchBP03.h>
#import <MicroLifeDeviceSDK/MicroLifeWatchBPOffice.h>
#import <MicroLifeDeviceSDK/MicroLifeWatchBPDRecord.h>
#import <MicroLifeDeviceSDK/MicroLifeWatchBPCurrentAndMData.h>
#import <MicroLifeDeviceSDK/MicroLifeWatchBPCBPdataAndCalCBP.h>
#import <MicroLifeDeviceSDK/MicroLifeWatchBPSettingValues.h>
#import <MicroLifeDeviceSDK/MicroLifeWatchBPFunctionSettingValues.h>
#import <MicroLifeDeviceSDK/MicroLifeWatchBPDDiagnosticDRecord.h>
#import <MicroLifeDeviceSDK/MicroLifeWatchBPNocturnalModeDRecord.h>
#import <MicroLifeDeviceSDK/MicroLifeWatchBPRemoteMeasurement.h>
#import <MicroLifeDeviceSDK/MicroLifeWatchBPM1.h>
```

3. SDK Bluetooth interface description:

1.14. Singletonization:

	+ (instancetype)shareOne;
Definition	Singleton

1.15. Instantiation:

	+ (instancetype)share;
Definition	Instantiate

1.16. Show Log :

	- (void)showLog:(BOOL)showLog;
Definition	Display SDK running Log
Parameter	showLog: whether to display Log

1.17. Set MicroLife Device :

	- (void)device:(NSArray *)bluetooth;
Definition	Set up the Bailuo device. When the mobile phone's Bluetooth is turned on, it will automatically search for surrounding devices, find a matching device, and automatically connect to the device by default.
Parameter	bluetooth: MicroLifeDevice Class
	<pre>[self.sdk device:@ [self.bp3g,self.bp4g,self.temperature,self.oxygen,self .weight,self.bloodSugar]];</pre>

1.18. Set Auto Scan :

	- (void)autoScan:(BOOL)autoScan;
Definition	When CBManagerStatePoweredOn automatically opens scanning [Default: Open]
Parameter	autoScan: Auto Scan

1.19. Set Sacn stop Time :

	- (void)stopTime:(NSInteger)time;
Definition	Set scan stop time
Parameter	time: scan stop time

1.20. Set RSSI :

	- (void)rssi:(NSInteger)rssi;
Definition	Set scanning RSSI strength
Parameter	rssi: Scan RSSI strength

1.21. Start Sacn :

	- (void)startSacn;
--	--------------------

Definition	Start Sacn
------------	------------

1.22. Cancel Scan :

	- (void)cancelScan;
Definition	Cancel Scan

1.23. Cancel All Connect :

	- (void)cancelAllConnect;
Definition	Cancel All Connect

1.24. Get the currently connected devices :

	- (NSArray *)findConnectedDevices;
Definition	Get the currently connected devices

1.25. When CentralManager state changed :

	- (void)getDidUpdateStateBlock:(didUpdateStateBlock)block;
Definition	Monitor the Bluetooth status of mobile phone
Parameter	void(^didUpdateStateBlock) (CBManagerState state)

1.26. Scan Device :

	- (void)getScanDeviceBlock:(scanDeviceBlock)scanDeviceBlock CancelScanBlock:(cancelScanBlock)cancelScanBlock;
Definition	Monitor scanning device status
Parameter	void(^scanDeviceBlock) (id device)
Parameter	void(^cancelScanBlock) (void)

1.27. Connect Device State :

	- (void)getConnectDeviceStateBlock:(connectDeviceStateBlock) connectDeviceStateBlock CancelAllDevicesConnectionBlock:(cancelAllDevicesConnectionBlock)cancelAllDevicesConnectionBlock;
Definition	Monitor the status of connected devices
Parameter	void(^connectDeviceStateBlock) (id device, CBCentralManager *central, CBPeripheral *peripheral, CBPeripheralState state, NSError * error)
Parameter	void(^cancelAllDevicesConnectionBlock) (void)

4.

第2章

MicroLifeDevice interface description:

2.1. Singletonization:

	+ (instancetype)shareWhithAuthorizationkey:(NSString *)key;
Definition	Singleton
Parameter	key: Authorization code

2.2. Add Device Model :

	- (void)addDeviceModel:(NSArray *)models;
Definition	Search for device custom models
Parameter	models: device name which can be included single or multiple
	<pre>[self.watchBP03 addDeviceModel:@[@"Watch BP 03"]];</pre>

2.3. Set Auto Scan :

	- (void)autoScan:(BOOL)autoScan;
Definition	When CBManagerStatePoweredOn automatically opens scanning [Default: Open]
Parameter	autoScan: Auto Scan

2.4. Auto Connect [Default: Open] :

	- (void)setAutoConnect:(BOOL)autoConnect;
Definition	Set Auto Connect [Default: Open]
Parameter	autoConnect: Auto Connect

2.5. ReConnect Time [Default: 0] :

	- (void)setReConnectTime:(NSInteger)reConnectIndex;
Definition	Set ReConnect Time [Default: 0], reconnection interval
Parameter	reConnectIndex: ReConnect Time

2.6. Set Sacn stop Time :

	- (void)stopTime:(NSInteger)time;
Definition	Set scan stop time, use Bluetooth independent management mode
Parameter	time: scan stop time

2.7. Set RSSI :

	- (void)rssi:(NSInteger)rssi;
Definition	Set scanning RSSI intensity, use Bluetooth independent management mode
Parameter	rssi: Scan RSSI strength

2.8. Start Sacn :

	- (void)startSacn;
Definition	Start Sacn, used in Bluetooth independent management mode

2.9. Cancel Scan :

	- (void)cancelScan;
Definition	Cancel Sacn, used in Bluetooth independent management mode

2.10. Connect :

	- (void)connectDevice;
Definition	Connect

2.11. Disconnect :

	- (void)disconnectDevice;
Definition	Disconnect

2.12. Set Auto Reconnect :

	- (void)setAutoReconnect;
Definition	Set up automatic reconnection and start by default

2.13. Cancel Auto Reconnect :

	- (void)cancelAutoReconnect;
Definition	Cancel automatic reconnection

2.14. Device Connected :

	- (void)deviceConnectedBlock:(deviceConnectedBlock)block;
Definition	The monitoring device has registered all preset services
Parameter	void(^deviceConnectedBlock) (void);

2.15. Device independent bluetooth :

	- (void)getDidUpdateStateBlock:(didUpdateStateBlock)didUpdateStateBlock ScanDeviceBlock:(scanDeviceBlock)scanDeviceBlock CancelScanBlock:(cancelScanBlock)cancelScanBlock ConnectDeviceStateBlock:(connectDeviceStateBlock)connectDeviceStateBlock CancelAllDevicesConnectionBlock:(cancelAllDevicesConnectionBlock)cancelAllDevicesConnectionBlock;
Definition	Use Bluetooth independent management mode to monitor mobile phone Bluetooth status, device scanning and connection status
Parameter	void(^didUpdateStateBlock) (CBManagerState state)

Parameter	void(^scanDeviceBlock) (id device)
Parameter	void(^cancelScanBlock) (void)
Parameter	void(^connectDeviceStateBlock) (id device, CBCentralManager *central, CBPeripheral *peripheral, CBPeripheralState state, NSError * error)
Parameter	void(^cancelAllDevicesConnectionBlock) (void)

2.16. Read RSSI :

	- (void)getReadRSSIBlock:(readRSSIBlock)block;
Definition	Monitor the RSSI status of the device.
Parameter	typedef void(^readRSSIBlock) (NSNumber *RSSI, NSError * error);

2.17. Read Data Value :

	- (void)getReadDataValueBlock:(readDataValueBlock)block;
Definition	Listening response status (original value).
Parameter	void(^readDataValueBlock) (NSInteger CMD, NSData * value, NSError * error);

2.18. Read Data Model :

	- (void)getReadDataModelBlock:(readDataModelBlock)block;
Definition	Monitor response status.
Parameter	void(^readDataModelBlock) (NSInteger CMD, id model, NSError * error);

2.19. Read Valify item Error :

	- (void)getValidationErrorBlock:(validationErrorBlock)block;
Definition	Monitor error response status.
Parameter	void(^validationErrorBlock) (NSString *item, NSString *info, NSData * value);

5. Temperature interface description

2.20. Temperature is active, and the SDK will only provide parsing functions.

2.21. Set the monitoring response status and obtain the corresponding data according to CMD

```
[self.temperature getReadDataModelBlock:^(NSInteger CMD, id
_Nonnull model, NSError * _Nonnull error) {
[weakSelf log:weakSelf.temperature CMD:CMD Model:model
Error:error];
switch (CMD) {
case CMDTemperatureReadDeviceInfo:
case CMDTemperatureReadMeasureData:
break;
default:
break;
}
}];
```

2.22. Read Device Info:

Definition	RRead Device Info
CMD	CMDTemperatureReadDeviceInfo
model	MicroLifeDeviceInfo

2.23. Read Measure Data:

Definition	Read Measure Data
CMD	CMDTemperatureReadMeasureData
model	MicroLifeTemperatureMeasureData

6. Blood Pressure interface description

- 2.24. Construct a corresponding manager according to the device type (3G/4G), set the monitoring response status, and obtain the corresponding data according to CMD

```
[self.bp3g getReadDataModelBlock:^(NSInteger CMD, id _Nonnull
model, NSError * _Nonnull error) {
[weakself log:weakself.bp3g CMD:CMD Model:model Error:error];
switch (CMD) {
case CMD3GReadHistorys:
case CMD3GClearAllHistorys:
case CMD3GReadUserAndVersionData:
case CMD3GWriteUser:
case CMD3GReadLastData:
case CMD3GClearLastData:
case CMD3GReadDeviceInfo:
break;
default:
break;
}
}];

[self.bp4g getReadDataModelBlock:^(NSInteger CMD, id _Nonnull
model, NSError * _Nonnull error) {
[weakself log:weakself.bp4g CMD:CMD Model:model Error:error];
switch (CMD) {
case CMD4GReadHistorys:
case CMD4GClearAllHistorys:
case CMD4GReadUserAndVersionData:
case CMD4GWriteUser:
case CMD4GReadDeviceInfo:
case CMD4GReadDeviceTime:
case CMD4GSyncTiming:
case CMD4GReadSerialNumber:
break;
default:
break;
}
}];
```

- 2.25. Read all history or current data from BPM :

	- (void)readAllHistorys;
Definition	Read all history or current data from BPM
CMD	CMD3GReadHistorys CMD4GReadHistorys
model	MicroLifeBloodPressureDRecord
	<pre>case CMD3GReadHistorys: [weakself BPMBLEManagerResponseReadHistory:model]; break;</pre>

	<pre>case CMD4GReadHistorys: [weakSelf BPMBLEManagerResponseReadHistory:model]; break;</pre>
--	--

2.26. Clear all history data of the BPM :

	- (void)clearAllHistorys;
Definition	clear all history data of the bpm
CMD	CMD3GClearAllHistorys CMD4GClearAllHistorys
model	MicroLifeDataModel
	<pre>case CMD3GClearAllHistorys: { MicroLifeDataModel *data = model; [weakSelf BPMBLEManagerResponseClearHistory:data.success .boolValue]; } break;</pre> <pre>case CMD4GClearAllHistorys: { MicroLifeDataModel *data = model; [weakSelf BPMBLEManagerResponseClearHistory:data.success .boolValue]; } break;</pre>

2.27. Disconnect the Bluetooth with BPM :

	- (void)disconnect;
Definition	Disconnect the Bluetooth with BPM
Parameter	void(^connectDeviceStateBlock) (id device, CBCentralManager *central, CBPeripheral *peripheral, CBPeripheralState state, NSError * error)
Parameter	void(^cancelAllDevicesConnectionBlock) (void)
	<pre>[self.sdk getConnectDeviceStateBlock:^(id _Nonnull device, CBCentralManager * _Nonnull central, CBPeripheral * _Nonnull peripheral, CBPeripheralState state, NSError * _Nonnull error) { [weakSelf connectDeviceInfo:device PeripheralState:state]; } CancelAllDevicesConnectionBlock:^([weakSelf addLogWhitText:@"Cancel All Devices Connection"];)];</pre>

2.28. Read user ID and version data from BPM :

	- (void)readUserAndVersionData;
Definition	Read user ID and version data from BPM: After the device is successfully connected, the user ID and version data will be automatically read.
CMD	CMD3GReadUserAndVersionData CMD4GReadUserAndVersionData
model	Dictionary : { UserInfo = "<MicroLifeUserInfo: User Information>";

	Version = "<MicroLifeDeviceInfo: Device Information>"; }
	<pre> case CMD3GReadUserAndVersionData: { MicroLifeUserInfo *userInfo = [model valueForKey:@"UserInfo"]; MicroLifeDeviceInfo *version = [model valueForKey:@"Version"]; [weakSelf BPMBLEManagerResponseReadUserAndVersionData:userInfo fo VersionData:version]; } break; case CMD4GReadUserAndVersionData: { MicroLifeUserInfo *userInfo = [model valueForKey:@"UserInfo"]; MicroLifeDeviceInfo *version = [model valueForKey:@"Version"]; [weakSelf BPMBLEManagerResponseReadUserAndVersionData:userInfo fo VersionData:version]; } break; </pre>

2.29. Write a new user ID to BPM :

	- (void)writeUserID:(NSString *)ID;
Definition	Write a new user ID to BPM
Parameter	ID : User ID.
CMD	CMD3GWriteUser CMD4GWriteUser
model	MicroLifeDataModel
	<pre> case CMD3GWriteUser: { MicroLifeDataModel *data = model; [weakSelf BPMBLEManagerResponseWriteUserID:data.success .boolValue]; } break; case CMD4GWriteUser: { MicroLifeDataModel *data = model; [weakSelf BPMBLEManagerResponseWriteUserID:data.success .boolValue]; } break; </pre>

2.30. Read device ID and info from BPM :

	- (void)readDeviceIDAndInfo;
Definition	Read device ID and info from BPM
CMD	CMD3GReadDeviceInfo CMD4GReadDeviceInfo
model	MicroLifeDeviceInfo

	<pre> case CMD3GReadDeviceInfo: [weakself BPMBLEManagerResponseReadDeviceInfo:model]; break; case CMD4GReadDeviceInfo: [weakself BPMBLEManagerResponseReadDeviceInfo:model]; </pre>
--	---

2.31. [4G Dedicated] Read device Time from BPM :

	- (void)readDeviceTime;
Definition	Read device Time from BPM
CMD	CMD4GReadDeviceTime
model	MicroLifeDeviceInfo
	<pre> case CMD4GReadDeviceTime: [weakself BPMBLEManagerResponseReadDeviceTime:model]; break; </pre>

2.32. [4G Dedicated] Write device Time to BPM :

	- (void)syncTiming;
Definition	Write device Time to BPM: After the device is successfully connected, the time will be automatically synchronized.
CMD	CMD4GSyncTiming
model	MicroLifeDataModel
	<pre> case CMD4GSyncTiming: { MicroLifeDataModel *data = model; [weakself BPMBLEManagerResponseWriteDeviceTime:data.success .boolValue]; } break; </pre>

2.33. Read last 1 data from the BPM : [Attention] will not respond!

	- (void)readLastData;(3G) - (BOOL)readLastData;(4G)
Definition	Read last 1 data from the BPM [Attention] will not respond!
CMD	CMD3GReadLastData CMD4GReadLastData
model	There is measurement data: MicroLifeBloodPressureDRecord No measurement data: MicroLifeDataModel

	<pre> case CMD3GReadLastData: { if ([model isKindOfClass:[MicroLifeBloodPressureDRecord class]]) { MicroLifeBloodPressureDRecord *dRecord = model; [weakSelf BPMBLEManagerResponseReadLastData: [dRecord.MData firstObject] HistoryMeasurementNumber:dRecord .historyMeasuremeNumber.intValue UserNumber:dRecord.userNumber.intValue MAMState:dRecord.MAMState.intValue IsNoData:YES]; } else { // reply Null ACK [weakSelf BPMBLEManagerResponseReadLastData:nil HistoryMeasurementNumber:nil UserNumber:nil MAMState:nil IsNoData:NO]; } } case CMD4GReadLastData: { if ([model isKindOfClass:[MicroLifeBloodPressureDRecord class]]) { MicroLifeBloodPressureDRecord *dRecord = model; [weakSelf BPMBLEManagerResponseReadLastData: [dRecord.MData firstObject] HistoryMeasurementNumber:dRecord .historyMeasuremeNumber.intValue UserNumber:dRecord.userNumber.intValue MAMState:dRecord.MAMState.intValue IsNoData:YES]; } else { // reply Null ACK [weakSelf BPMBLEManagerResponseReadLastData:nil HistoryMeasurementNumber:nil UserNumber:nil MAMState:nil IsNoData:NO]; } } </pre>
--	---

2.34. Clear last 1 data of the BPM : [Attention] will not respond!

	<ul style="list-style-type: none"> - (void)readLastData;(3G) - (BOOL)readLastData;(4G)
Definition	Clear last 1 data of the BPM : [Attention] will not respond!
CMD	CMD3GClearLastData CMD4GClearLastData
model	MicroLifeDataModel
	<pre> case CMD3GClearLastData: { MicroLifeDataModel *data = model; [weakSelf BPMBLEManagerResponseClearLastData:data.success .boolValue]; } break; </pre>

	<pre> case CMD4GClearLastData: { MicroLifeDataModel *data = model; [weakSelf BPMBLEManagerResponseClearLastData:data.success .boolValue]; } break; </pre>
--	---

2.35. Read device SN from BPM :

	- (BOOL)readSerialNumber;
Definition	Read device SN from BPM
CMD	CMD4GReadSerialNumber
model	MicroLifeDeviceInfo
	<pre> case CMD4GReadSerialNumber: { MicroLifeDeviceInfo *deviceInfo = model; NSLog(@"Serial Number:%@",deviceInfo.sn); } break; </pre>

7.Weight interface description:

- 2.36. Set the monitoring response status and obtain the corresponding data according to CMD

```

[weakSelf log:weakSelf.weight CMD:CMD Model:model Error:error];
}
}];

```

- 2.37. Weight WakeUp Scale :

Definition	Weight WakeUp Scale
CMD	CMDWeightWakeUpScale
	<pre> case CMDWeightWakeUpScale: [weakSelf addLogWhitText:@"WakeUp Scale"]; [weakSelf writeUserData]; [weakSelf.weight readHistorys]; break; </pre>

- 2.38. Weight Sleep Scale :

Definition	Weight Sleep Scale
CMD	CMDWeightSleepScale
	<pre> case CMDWeightSleepScale: [weakSelf addLogWhitText:@"Sleep Scale"]; [weakSelf.weight disconnect]; break; </pre>

- 2.39. Write User :

	- (void)writeUserData:(NSString *)ID Age:(NSInteger)age Gender:(MicroLifeUserGender)gender Height:(NSInteger)height RoleType:(MicroLifeUserRoleType)roleType Weight:(float)weight Resistance:(NSInteger)resistance;
Definition	Write User

	Note: The SDK also sends user information to the scale, so you may receive this callback message frequently
CMD	CMDWeightUpdateUserInfo
	<pre>case CMDWeightUpdateUserInfo: [weakSelf addLogWhitText:@"Update UserInfo"]; break;</pre>

2.40. No More OfflineData :

Definition	No More OfflineData
CMD	CMDWeightNoMoreOfflineData
	<pre>case CMDWeightNoMoreOfflineData: [weakSelf addLogWhitText:@"No More OfflineData"]; break;</pre>

2.41. Low Power :

Definition	Low Power
CMD	CMDWeightLowPower
	<pre>case CMDWeightLowPower: [weakSelf addLogWhitText:@"Low Power"]; break;</pre>

2.42. Sync System Clock :

Definition	Sync System Clock
CMD	CMDWeightSyncSystemClock
model	MicroLifeDeviceInfo
	<pre>case CMDWeightSyncSystemClock: [weakSelf addLogWhitText:@"Sync System Clock"]; break;</pre>

2.43. Clear All Users :

	- (void)clearAllUsers;
Definition	Clear All Users
CMD	CMDWeightClearAllUserInfo
	<pre>case CMDWeightClearAllUserInfo: [weakSelf addLogWhitText:@"Clear All UserInfo"]; break;</pre>

2.44. Read all history data :

	- (void)readHistorys;
Definition	Read all history data
CMD	CMDWeightReadHistorys
model	MicroLifeBodyFat

	<pre> case CMDWeightReadHistorys: [weakself addLogWhitText:@"Read Historys"]; weakself.currUser.resistance = ((MicroLifeBodyFat *)model).resistance; [weakself saveNSUserDefaults:weakself.currUser.resistance Key:@"resistance"]; break; </pre>
--	--

2.45. Read current data :

Definition	Read current data
CMD	CMDWeightMeasurementResult
model	MicroLifeBodyFat
	<pre> case CMDWeightMeasurementResult: [weakself addLogWhitText:@"Measurement Result"]; weakself.currUser.resistance = ((MicroLifeBodyFat *)model).resistance; [weakself saveNSUserDefaults:weakself.currUser.resistance Key:@"resistance"]; break; </pre>

2.46. Disconnect the Bluetooth with BPM :

	- (void)disconnect;
Definition	Disconnect the Bluetooth with BPM
Parameter	void(^connectDeviceStateBlock) (id device, CBCentralManager *central, CBPeripheral *peripheral, CBPeripheralState state, NSError * error)
Parameter	void(^cancelAllDevicesConnectionBlock) (void)
	<pre> [self.sdk getConnectDeviceStateBlock:^(id _Nonnull device, CBCentralManager * _Nonnull central, CBPeripheral * _Nonnull peripheral, CBPeripheralState state, NSError * _Nonnull error) { [weakself connectDeviceInfo:device PeripheralState:state]; } CancelAllDevicesConnectionBlock:^([weakself addLogWhitText:@"Cancel All Devices Connection"]; [weakself.weight startScan]; }]; </pre>

2.47. Body fat machine body composition analysis :

2.47.1. Just import the header file.

```
#import <BelterScaleInfo/BelterScaleInfo.h>
```

2.47.2. initialization:

	- (void)getInstance;
Definition	Instance
Parameter	
Parameter	

2.47.3. Set user information :

	- (void)setUserInfo_height:(double)height weight:(double)weight resistance:(int)resistance sex:(int)sex age:(int)age csRoleType:(BTRoleType)typ;
Definition	Set user information
Parameter	- (double)getMoisture; - (double)getMuscle; - (double)getProtein; - (double)getBMR; - (double)getBone; - (double)getVisceralfat; - (double)getBoneMuscle; - (double)getBMI; - (int)getBodyAge; - (int)getBodyScore;
	<pre> int sex = (self.currUser.gender == MicroLifeUserGenderFemale)?0:1; BTRoleType type = (self.currUser.roleType == MicroLifeUserRoleTypeNormal)?BTRoleTypeNormal:BTRoleTypeSportsman; BelterScaleInfo *scaleInfo = [BelterScaleInfo getInstance]; [scaleInfo setUserInfo_height:self.currUser.height.doubleValue weight:bodyFat.weight.doubleValue resistance:bodyFat.resistance.intValue sex:sex age:self.currUser.age.intValue csRoleType:type]; bodyFat.bmi = @(scaleInfo.getBMI); bodyFat.visceralfat = @(scaleInfo.getVisceralfat); bodyFat.bmr = @(scaleInfo.getBMR); bodyFat.muscle = @(scaleInfo.getMuscle); bodyFat.protein = @(scaleInfo.getProtein); bodyFat.bone = @(scaleInfo.getBone); bodyFat.moisture = @(scaleInfo.getMoisture); bodyFat.boneMuscle = @(scaleInfo.getBoneMuscle); bodyFat.bodyScore = @(scaleInfo.getBodyScore); bodyFat.bodyAge = @(scaleInfo.getBodyAge); </pre>

8.Oxygen interface description

- 2.48. Set the monitoring response status and obtain the corresponding data according to CMD

```
[self.oxygen getReadDataModelBlock:^(NSInteger CMD, id _Nonnull
model, NSError * _Nonnull error) {
[weakself log:weakself.oxygen CMD:CMD Model:model Error:error];
switch (CMD) {
case CMDOxygenVolumeTracingData:
[weakself showOxygen:model];
break;
case CMDOxygenSaturationAndPulseRateData:
break;
case CMDOxygenSaturationAndPulseRateAlarmLimits:
break;
default:
break;
}
}];
```

- 2.49. Read Oxygen Volume Tracing Data :

Definition	Oxygen Volume Tracing Data
CMD	CMDOxygenVolumeTracingData
model	MicroLifeOxygenData
	<pre>case CMDOxygenVolumeTracingData: [weakself showOxygen:model]; break;</pre>

- 2.50. Read blood oxygen saturation and pulse rate alarm limit :

	-(void)readAlarmInfo;
Definition	Read blood oxygen saturation and pulse rate alarm limit
CMD	CMDOxygenSaturationAndPulseRateData
model	MicroLifeOxygenData
	<pre>case CMDOxygenSaturationAndPulseRateData: break;</pre>

- 2.51. Write blood oxygen saturation and pulse rate alarm limit :

	-(void)alarmSPOMax:(NSInteger)maxSPO SPOmin:(NSInteger)minSPO PlusMax:(NSInteger)maxPlus Plusmin:(NSInteger)minPlus;
Definition	Write blood oxygen saturation and pulse rate alarm limit
Parameter	maxSPO The lower limit of blood oxygen alarm limit, the value is 50-100. minSPO The upper limit of blood oxygen alarm limit, the value is 50-100.

	maxPlus The lower limit of the pulse rate alarm limit, the value is 5-250. It must be a multiple of 5 and cannot be equal to 0. minPlus The upper limit of the pulse rate alarm limit, the value is 5-250, must be a multiple of 5, and cannot be equal to 0.
CMD	CMDOxygenSaturationAndPulseRateAlarmLimits
model	MicroLifeOxygenData
	<pre>case CMDOxygenSaturationAndPulseRateAlarmLimits: break;</pre>

9.WatchBP O3 interface description

- 2.52. Set the monitoring response status and obtain the corresponding data according to CMD

```
[self.watchBP03 getReadDataModelBlock:^(NSInteger CMD, id
_Nonnull model, NSError * _Nonnull error) {
    switch (CMD) {
        case CMDWatchBP03ReadCBPData:
        case CMDWatchBP03ReadDeviceInfo:
        case CMDWatchBP03ReadDeviceTime:
        case CMDWatchBP03ReadSerialNumber:
        case CMDWatchBP03ReadBTModuleName:
        case CMDWatchBP03ClearAllHistorys:
        case CMDWatchBP03WriteUser:
        case CMDWatchBP03WriteSettingValues:
        case CMDWatchBP03ReadFunctionSettingValue:
        case CMDWatchBP03WriteDeviceTime:
        case CMDWatchBP03ReadSettingValues:
        case CMDWatchBP035SReadSettingValues:
        case CMDWatchBP03ReadHistorys:
        case CMDWatchBP03ReadUserAndVersionData:
            break;
        default:
            break;
    }
}
```

- 2.53. Read all history or current data from BPM :

	- (void)readAllHistorys;
Definition	Read all history or current data from BPM
CMD	CMDWatchBP03ReadHistorys
model	There is measurement data: MicroLifeWatchBPRecord No measurement data: MicroLifeDataModel
	<pre>case CMDWatchBP03ReadHistorys: if ([model isKindOfClass:[MicroLifeWatchBPRecord class]]) { [weakSelf WB03BLEManagerResponseReadAllHistorys:model]; } else { // reply Null ACK } break;</pre>

- 2.54. Read CBP data by index from BPM :

	- (void)readCBPDataWithIndex:(int)index Dformat:(Dformat)dformat;
Definition	Read diagnostic mode history data from BPM

Parameter	<p>index : is CBP memory index</p> <p>dformat : Data format which BPM sent out. It is same as what APP requested.</p> <p>NoCBPRaw : No CBP raw data</p> <p>LowCBPRaw : low resolution CBP data (sampling rate =16Hz)</p> <p>FullCBPRaw : full CBP raw data (sampling rate=256Hz)</p>
CMD	CMDWatchBPO3ReadCBPData
model	<p>There is measurement data: MicroLifeWatchBPCBPdataAndCalCBP</p> <p>No measurement data: MicroLifeDataModel</p>
	<pre> case CMDWatchBPO3ReadCBPData: if ([model isKindOfClass:[MicroLifeWatchBPCBPdataAndCalCBP class]]) { [weakSelf WB03BLEManagerResponseReadCBPData:model IsNoData:NO]; } else { [weakSelf WB03BLEManagerResponseReadCBPData:nil IsNoData:YES]; } break; </pre>

2.55. Clear all history data of the BPM :

	- (void)clearAllHistorys;
Definition	clear all history data of the bpm
CMD	CMDWatchBPO3ClearAllHistorys
model	MicroLifeDataModel
	<pre> case CMDWatchBPO3ClearAllHistorys: { MicroLifeDataModel *dataModel = model; [weakSelf WB03BLEManagerResponseClearHistory:dataModel .success.boolValue]; } break; </pre>

2.56. Disconnect the Bluetooth with BPM :

	- (void)disconnect;
Definition	Disconnect the Bluetooth with BPM
Parameter	void(^connectDeviceStateBlock) (id device, CBCentralManager *central, CBPeripheral *peripheral, CBPeripheralState state, NSError * error)
Parameter	void(^cancelAllDevicesConnectionBlock) (void)

	<pre>// Device connection status response [self.sdk getConnectDeviceStateBlock:^(id _Nonnull device, CBCentralManager * _Nonnull central, CBPeripheral * _Nonnull peripheral, CBPeripheralState state, NSError * _Nonnull error) { weakself.setting2g.hidden = weakself.setting5g.hidden = YES; [weakself connectDeviceInfo:device PeripheralState:state]; } CancelAllDevicesConnectionBlock:^([weakself addLogWhitText:@"Cancel All Devices Connection"]; }];</pre>
--	---

2.57. Read user ID and version data from BPM :

	- (void)readUserAndVersionData;
Definition	Read user ID and version data from BPM: After the device is successfully connected, the user ID and version data will be automatically read.
CMD	CMDWatchBPO3ReadUserAndVersionData
model	Dictionary : { UserInfo = "<MicroLifeUserInfo: User Information>"; Version = "<MicroLifeDeviceInfo: Device Information>"; }
	<pre>case CMDWatchBPO3ReadUserAndVersionData: { MicroLifeUserInfo *userInfo = [model valueForKey:@"UserInfo"]; MicroLifeDeviceInfo *version = [model valueForKey:@"Version"]; [weakself WB03BLEManagerResponseReadUserAndVersionData:userInfo VersionData:version]; } break;</pre>

2.58. Write a new user ID to BPM :

	- (void)writeUserID:(NSString *)ID;
Definition	Write a new user ID to BPM
Parameter	ID : User ID.
CMD	CMDWatchBPO3WriteUser
model	MicroLifeDataModel
	<pre>case CMDWatchBPO3WriteUser: { MicroLifeDataModel *dataModel = model; [weakself WB03BLEManagerResponseWriteUserID:dataModel .success.boolValue]; } break;</pre>

2.59. Read ABPM setting values from BPM :

	- (void)readSettingValues;
Definition	Read ABPM setting values from BPM
CMD	CMDWatchBPO3ReadSettingValues CMDWatchBPO35SReadSettingValues

model	MicroLifeWatchBPSettingValues
	<pre> case CMDWatchBP03ReadSettingValues: case CMDWatchBP035SReadSettingValues: [weakSelf WB03BLEManagerResponseReadSettingValue:model]; break; </pre>

2.60. Write ABPM setting values to BPM :

	<p>-</p> <p>(BOOL)writeSettingValuesWithSelectedABPMStart:(NSInteger) ABPMStart ABPMEnd:(NSInteger)ABPMEnd ABPMInt_first:(MeasurementTime)ABPMInt_first ABPMInt_second:(MeasurementTime)ABPMInt_second HI_infPressure:(HlInfPressure)HI_infPressure SW_checkhide:(BOOL)SW_checkhide SW_SEL_silent:(BOOL)SW_SEL_silent CBP_zone1_meas_off:(BOOL)CBP_zone1_meas_off CBP_zone2_meas_off:(BOOL)CBP_zone2_meas_off CBPInt_first:(MeasurementTime)CBPInt_first CBPInt_second:(MeasurementTime)CBPInt_second;</p>
Definition	Write ABPM setting values to BPM.
Parameter	<p>ABPMStart The starting time of the first measurement time zone</p> <p>ABPMEnd The end time of the first measurement time zone</p> <p>ABPMInt_first The interval of the first measurement time zone (valid range: 5, 10, 15, 20, 30, and 60)</p> <p>ABPMInt_second The interval of the second measurement time zone (valid range: 5, 10, 15, 20, 30, and 60)</p> <p>HI_infPressure Highest inflation pressure of Auto mode</p> <p>SW_checkhide The interval of the CBP first measurement time zone Note: CBPInt_first should multiple time than ABPMInt_first.</p> <p>SW_SEL_silent Hide(true)/Show(false) readings after measurement</p> <p>CBP_zone1_meas_off the first time zone of CBP measurement true:disabled/false:enabled</p> <p>CBP_zone2_meas_off the second time zone of CBP measurement true:disabled/false:enabled</p> <p>CBPInt_first The interval of the CBP first measurement time zone (valid range: 5, 10, 15, 20, 30, and 60) Note: CBPInt_first should multiple time than ABPMInt_first.</p> <p>CBPInt_second The interval of the CBP second measurement time zone (valid range: 5, 10, 15, 20, 30, and 60) Note: CBPInt_second should multiple time than ABPMInt_second.</p>
CMD	CMDWatchBP03WriteSettingValues
model	MicroLifeDataModel

	<pre>[self.watchBP03 writeSettingValuesWithSelectedABPMStart:self.settingValue .ABPMStart.integerValue ABPMEnd:self.settingValue.ABPMEnd.integerValue ABPMInt_first:self.settingValue.ABPMInt_first ABPMInt_second:self.settingValue.ABPMInt_second HI_infPressure:self.settingValue.HI_infPressure SW_checkhide:self.settingValue.SW_checkhide.boolValue SW_SEL_silent:self.settingValue.SW_SEL_silent.boolValue CBP_zone1_meas_off:self.settingValue.CBP_zone1_meas_off .boolValue CBP_zone2_meas_off:self.settingValue.CBP_zone2_meas_off .boolValue CBPInt_first:self.settingValue.CBPInt_first CBPInt_second:self.settingValue.CBPInt_second];</pre>
	<pre>case CMDWatchBP03WriteSettingValues: case CMDWatchBP035SWriteSettingValues: { MicroLifeDataModel *dataModel = model; [weakSelf WB03BLEManagerResponseWriteSettingValues:dataModel .success.boolValue]; } break;</pre>

2.61. Write Setting Values from BPM(5 schedule) :

	<pre>- (BOOL)writeSettingValuesWithSettingValues:(NSMutableArray *)settingValues HI_infPressure:(HI_infPressure)HI_infPressure SW_checkhide:(BOOL)SW_checkhide;</pre>
Definition	Write Setting Values from BPM(5 schedule)
Parameter	settingValues MicroLifeWatchBPSettingValues HI_infPressure HI_infPressure SW_checkhide SW_checkhide
CMD	CMDWatchBP035SWriteSettingValues
model	MicroLifeDataModel
	<pre>[self.watchBP03 writeSettingValuesWithSettingValues:self .settingValue.settingValues HI_infPressure:self.settingValue .HI_infPressure SW_checkhide:self.settingValue .SW_checkhide.boolValue];</pre>
	<pre>case CMDWatchBP03WriteSettingValues: case CMDWatchBP035SWriteSettingValues: { MicroLifeDataModel *dataModel = model; [weakSelf WB03BLEManagerResponseWriteSettingValues:dataModel .success.boolValue]; } break;</pre>

2.62. Read device ID and info from BPM :

	- (void)readDeviceIDAndInfo;
Definition	Read device ID and info from BPM

CMD	CMDWatchBPO3ReadDeviceInfo
model	MicroLifeDeviceInfo
	<pre> case CMDWatchBPO3ReadDeviceInfo: [weakSelf WB03BLEManagerResponseReadDeviceIDAndInfo:model]; break; </pre>

2.63. Read device Time from BPM :

	- (void)readDeviceTime;
Definition	Read device Time from BPM
CMD	CMDWatchBPO3ReadDeviceTime
model	MicroLifeDeviceInfo
	<pre> case CMDWatchBPO3ReadDeviceTime: [weakSelf WB03BLEManagerResponseReadDeviceTime:model]; break; </pre>

2.64. Write device Time to BPM :

	- (void)writeDeviceTime;
Definition	Write device Time to BPM: After the device is successfully connected, the time will be automatically synchronized.
CMD	CMDWatchBPO3WriteDeviceTime
model	MicroLifeDataModel
	<pre> case CMDWatchBPO3WriteDeviceTime: { MicroLifeDataModel *dataModel = model; [weakSelf WB03BLEManagerResponseWriteDeviceTime:dataModel .success.boolValue]; } break; </pre>

2.65. Read BPM function setting value from BPM :

	- (void)readFunctionSettingValue;
Definition	Read BPM function setting value from BPM
CMD	CMDWatchBPO3ReadFunctionSettingValue
model	MicroLifeWatchBPFFunctionSettingValues
	<pre> case CMDWatchBPO3ReadFunctionSettingValue: [weakSelf WB03BLEManagerResponseReadFunctionSettingValue:mod el]; break; </pre>

2.66. Read BT module name from BPM :

	- (void)readBTModuleName;
Definition	Read BT module name from BPM
CMD	CMDWatchBPO3ReadBTModuleName
model	MicroLifeDeviceInfo

	<pre> case CMDWatchBP03ReadBTModuleName: { MicroLifeDeviceInfo *deviceInfo = model; [weakSelf WB03BLEManagerResponseReadBTModuleName:deviceInfo .BTModuleName]; } break; </pre>
--	---

2.67. Read device SN from BPM :

	- (void)readSerialNumber;
Definition	Read device SN from BPM
CMD	CMDWatchBP03ReadSerialNumber
model	MicroLifeDeviceInfo
	<pre> case CMDWatchBP03ReadSerialNumber: { MicroLifeDeviceInfo *deviceInfo = model; NSLog(@"Serial Number:%@",deviceInfo.sn); } break; </pre>

10.WatchBP Home interface description

- 2.68. Set the monitoring response status and obtain the corresponding data according to CMD

```
[self.watchBPHome getReadDataModelBlock:^(NSInteger CMD, id _Nonnull model, NSError * _Nonnull error) {
    [weakSelf log:weakSelf.watchBPHome CMD:CMD Model:model Error:error];
    switch (CMD) {
        case CMDWatchBPHomeReadUsualModeHistoryData:
        case CMDWatchBPHomeReadUsualModeHistoryDataIncludeEachMeasurement:
        case CMDWatchBPHomeReadDiagnosticModeHistoryData:
        case CMDWatchBPHomeReadNocturnalModeSetting:
        case CMDWatchBPHomeReadDeviceInfo:
        case CMDWatchBPHomeReadDeviceTime:
        case CMDWatchBPHomeReadSerialNumber:
        case CMDWatchBPHomeReadUserAndVersionData:
        case CMDWatchBPHomeReadNocturnalModeHistoryData:
        case CMDWatchBPHomeClearHistoryDataMode:
        case CMDWatchBPHomeClearCurrentModeHistoryData:
        case CMDWatchBPHomeWriteDeviceTime:
        case CMDWatchBPHomeWriteUserID:
        case CMDWatchBPHomeChangeNocturnalMode:
        case CMDWatchBPHomeReadDeviceSettingOfMeasurement:
        case CMDWatchBPHomeWriteDeviceSettingOfMeasurement:
            break;
        default:
            break;
    }
}];
```

- 2.69. Read usual mode history data from BPM :

	- (void)readUsualModeHistoryData;
Definition	Read all history or current data from BPM
CMD	CMDWatchBPHomeReadUsualModeHistoryData
model	There is measurement data: MicroLifeWatchBPRecord No measurement data: MicroLifeDataModel
	<pre>case CMDWatchBPHomeReadUsualModeHistoryData: { if ([model isKindOfClass:[MicroLifeWatchBPRecord class]]) { [weakSelf WBPBLEManagerResponseReadUsualModeHistoryData: model IsNoData:NO]; } else { [weakSelf WBPBLEManagerResponseReadUsualModeHistoryData: nil IsNoData:YES]; } } break;</pre>

- 2.70. Read diagnostic mode history data from BPM :

	- (void)readDiagnosticModeHistoryData;
Definition	Read diagnostic mode history data from BPM
CMD	CMDWatchBPHomeReadDiagnosticModeHistoryData
model	There is measurement data: MicroLifeWatchBPDiagnosticDRecord No measurement data: MicroLifeDataModel

	<pre> case CMDWatchBPHomeReadDiagnosticModeHistoryData: { if ([model isKindOfClass:[MicroLifeWatchBPDDiagnosticDRecord class]]) { [weakSelf WBPBLEManagerResponseReadDiagnosticModeHistory Data:model IsNoData:NO]; } else { [weakSelf WBPBLEManagerResponseReadDiagnosticModeHistory Data:nil IsNoData:YES]; } } break; </pre>
--	---

2.71. Clear selected mode history data of the BPM :

	- (void)clearHistoryDataWithSelectedUsualMode:(BOOL)clearUsualMode DiagnosticMode:(BOOL)clearDiagnosticMode NocturnalMode:(BOOL)clearNocturnalMode;
Definition	Clear selected mode history data of the BPM
Parameter	clearUsualMode clear Usual Mode clearDiagnosticMode clear Diagnostic Mode clearNocturnalMode clear Nocturnal Mode
CMD	CMDWatchBPHomeClearHistoryDataMode
model	MicroLifeDataModel
	<pre> case CMDWatchBPHomeClearHistoryDataMode: { MicroLifeDataModel *dataModel = model; [weakSelf WBPBLEManagerResponseClearCurrentModeHistoryData:d ataModel.success.boolValue]; } break; </pre>

2.72. Clear current mode history data of the BPM :

	- (void)clearCurrentModeHistoryData;
Definition	Clear current mode history data of the BPM
CMD	CMDWatchBPHomeClearCurrentModeHistoryData
model	MicroLifeDataModel
	<pre> case CMDWatchBPHomeClearCurrentModeHistoryData: { MicroLifeDataModel *dataModel = model; [weakSelf WBPBLEManagerResponseClearCurrentModeHistoryData:d ataModel.success.boolValue]; } break; </pre>

2.73. Disconnect the Bluetooth with BPM :

	- (void)disconnect;
Definition	Disconnect the Bluetooth with BPM

Parameter	void(^connectDeviceStateBlock) (id device, CBCentralManager *central, CBPeripheral *peripheral, CBPeripheralState state, NSError * error)
Parameter	void(^cancelAllDevicesConnectionBlock) (void)
	<pre> [self.sdk getConnectDeviceStateBlock:^(id _Nonnull device, CBCentralManager * _Nonnull central, CBPeripheral * _Nonnull peripheral, CBPeripheralState state, NSError * _Nonnull error) { [weakself connectDeviceInfo:device PeripheralState:state]; } CancelAllDevicesConnectionBlock:^([weakself addLogWhitText:@"Cancel All Devices Connection"];)]; </pre>

2.74. Read user ID and version data from BPM :

	- (void)readUserAndVersionData;
Definition	Read user ID and version data from BPM: After the device is successfully connected, the user ID and version data will be automatically read.
CMD	CMDWatchBPHomeReadUserAndVersionData
model	Dictionary : { UserInfo = "<MicroLifeUserInfo: User Information>"; Version = "<MicroLifeDeviceInfo: Device Information>"; }
	<pre> case CMDWatchBPHomeReadUserAndVersionData: { MicroLifeUserInfo *userInfo = [model valueForKey:@"UserInfo"]; MicroLifeDeviceInfo *version = [model valueForKey:@"Version"]; [weakself WBPBLEManagerResponseReadUserAndVersionData:userInfo VersionData:version]; } break; </pre>

2.75. Write a new user ID to BPM :

	- (void)writeUserID:(NSString *)ID;
Definition	Write a new user ID to BPM
Parameter	ID : User ID.
CMD	CMDWatchBPHomeWriteUserID
model	MicroLifeDataModel
	<pre> case CMDWatchBPHomeWriteUserID: { MicroLifeDataModel *dataModel = model; [weakself WBPBLEManagerResponseWriteUserID:dataModel .success.boolValue]; } break; </pre>

2.76. Read ABPM setting values from BPM :

	- (void)readSettingValues;
--	----------------------------

Definition	Read ABPM setting values from BPM
CMD	CMDWatchBPO3ReadSettingValues CMDWatchBPO35SReadSettingValues
model	MicroLifeWatchBPSettingValues
	<pre> case CMDWatchBPO3ReadSettingValues: case CMDWatchBPO35SReadSettingValues: [weakSelf WBO3BLEManagerResponseReadSettingValue:model]; break; </pre>

2.77. Read device ID and info from BPM :

	- (void)readDeviceIDAndInfo;
Definition	Read device ID and info from BPM
CMD	CMDWatchBPHomeReadDeviceInfo
model	MicroLifeDeviceInfo
	<pre> case CMDWatchBPHomeReadDeviceInfo: [weakSelf WBPBLEManagerResponseReadDeviceIDAndInfo:model]; break; </pre>

2.78. Read device Time from BPM :

	- (void)readDeviceTime;
Definition	Read device Time from BPM
CMD	CMDWatchBPHomeReadDeviceTime
model	MicroLifeDeviceInfo
	<pre> case CMDWatchBPHomeReadDeviceTime: [weakSelf WBPBLEManagerResponseReadDeviceTime:model]; break; </pre>

2.79. Write device Time to BPM :

	- (void)writeDeviceTime;
Definition	Write device Time to BPM: After the device is successfully connected, the time will be automatically synchronized.
CMD	CMDWatchBPHomeWriteDeviceTime
model	MicroLifeDataModel
	<pre> case CMDWatchBPHomeWriteDeviceTime: { MicroLifeDataModel *dataModel = model; [weakSelf WBPBLEManagerResponseWriteDeviceTime:dataModel .success.boolValue]; } break; </pre>

2.80. Read nocturnal mode setting :

	- (void)readNocturnalModeSetting;
Definition	Read nocturnal mode setting
CMD	CMDWatchBPHomeReadNocturnalModeSetting
model	MicroLifeDeviceInfo

	<pre> case CMDWatchBPHomeReadNocturnalModeSetting: [weakSelf WBPBLEManagerResponseReadNocturnalModeSetting:mode]; break; </pre>
--	--

2.81. Change nocturnal mode setting :

	- (void)changeNocturnalModeSettingOn:(BOOL)open StartYear:(NSInteger)year StartMonth:(NSInteger)month StartDay:(NSInteger)day StartHour:(NSInteger)hour;
Definition	Change nocturnal mode setting Note : This command requires matching hardware to set. You can use "readDeviceIDAndInfo(MicroLifeDeviceInfo.openNocturnalMode)" to check if the device supports it.
Parameter	open Nocturnal ON/OFF year year month month day day hour hour
CMD	CMDWatchBPHomeChangeNocturnalMode
model	MicroLifeDataModel
	<pre> case CMDWatchBPHomeChangeNocturnalMode: { MicroLifeDataModel *dataModel = model; [weakSelf WBPBLEManagerResponseChangeNocturnalModeSetting:da taModel.success.boolValue]; } break; </pre>

2.82. Read Nocturnal mode history data from BPM :

	- (void)readNocturnalModeHistoryData;
Definition	Read Nocturnal mode history data from BPM,If memory bank is blank, BPM will respond NullACK
CMD	CMDWatchBPHomeReadNocturnalModeHistoryData
model	There is measurement data: MicroLifeWatchBPNocturnalModeDRecord No measurement data: MicroLifeDataModel
	<pre> case CMDWatchBPHomeReadNocturnalModeHistoryData: { if ([model isKindOfClass: [MicroLifeWatchBPNocturnalModeDRecord class]]) { [weakSelf WBPBLEManagerResponseReadNocturnalPatternHisto ryData:model IsNoData:NO]; } else { [weakSelf WBPBLEManagerResponseReadNocturnalPatternHisto ryData:nil IsNoData:YES]; } } break; </pre>

2.83. Read device SN from BPM :

	- (void)readSerialNumber;
Definition	Read device SN from BPM
CMD	CMDWatchBPHomeReadSerialNumber
model	MicroLifeDeviceInfo
	<pre> case CMDWatchBPHomeReadSerialNumber: [weakSelf WBPBLEManagerResponseReadSerialNumber:model]; break; </pre>

2.84. Read measurement setting :

	- (BOOL)readDeviceSettingOfMeasurement
Definition	Read measurement setting
CMD	CMDWatchBPHomeReadDeviceSettingOfMeasurement
model	MicroLifeDeviceInfo
	<pre> case CMDWatchBPHomeReadDeviceSettingOfMeasurement: { MicroLifeDeviceInfo *deviceInfo = model; NSLog(@"Read Device Setting Of Measurement:%@",deviceInfo.parseDictionary); } break; </pre>

2.85. Write measurement setting :

	<p>-</p> <p>(BOOL)writeDeviceSettingOfMeasurement:(MeasurementTimes)measurementTimes RestTime:(NSInteger)restTime IntervalTime:(NSInteger)intervalTime ExcludeAverage:(ExcludeAverage)excludeAverage SW_Afib:(BOOL)SW_Afib;</p>
Definition	Write measurement setting
Parameter	<p>measurementTimes Measurement times: (Default =3) It's the measurement times (1~3) for usual mode.</p> <p>restTime Rest time: (Default = 0) It's interval before 1st measurement.</p> <p>intervalTime Interval time: (Default = 15) It's interval between each measurement.</p> <p>excludeAverage Exclude average: (Default = 0)</p> <p>0: average all measurements</p> <p>1: average excludes 1st measurement</p> <p>SW_Afib SW_Afib enable the Afib function. (this setting is valid with the Afib option enable in "BPMSetting2".)</p>
CMD	CMDWatchBPHomeWriteDeviceSettingOfMeasurement
model	MicroLifeDataModel

	<pre> case CMDWatchBPHomeReadDeviceSettingOfMeasurement: { MicroLifeDeviceInfo *deviceInfo = model; NSLog(@"Read Device Setting Of Measurement:%@",deviceInfo.parseDictionary); } break; case CMDWatchBPHomeWriteDeviceSettingOfMeasurement: { MicroLifeDataModel *dataModel = model; NSLog(@"Write Device Setting Of Measurement:%@",dataModel.success); } break; </pre>
--	---

2.86. Read usual mode history data (include each measurement) from
BPM :

	-
	(BOOL)readUsualModeHistoryDataIncludeEachMeasurement;
Definition	Read usual mode history data (include each measurement)
CMD	CMDWatchBPHomeReadUsualModeHistoryDataIncludeEach Measurement
model	MicroLifeWatchBPRecord
	<pre> case CMDWatchBPHomeReadUsualModeHistoryDataIncludeEachMeasurement: { MicroLifeWatchBPRecord *dRecord = model; NSString *log = [NSString stringWithFormat:@"dRecord historyMeasuremeNumber:%@ MData:%ld",dRecord.historyMeasuremeNumber,dRecord.MData.count]; [self addLogWhitText:log]; } break; </pre>

11. WatchBP Office interface description

- 2.87. Set the monitoring response status and obtain the corresponding data according to CMD

```
[self.watchBPOffice getReadDataModelBlock:^(NSInteger
    CMD, id _Nonnull model, NSError * _Nonnull error) {
    [weakself log:weakself.watchBPOffice CMD:CMD
        Model:model Error:error];
    switch (CMD) {
        case CMDWatchBPOfficeReadHistorys:
            [weakself
                WBOBLEManagerResponseReadAllHistorys:model
            ];
            break;
        case CMDWatchBPOfficeReadUserAndVersionData: {
            MicroLifeUserInfo *userInfo = [model
                valueForKey:@"UserInfo"];
            weakself.editUserID.text = userInfo.bpmUserID;
        }
            break;
        case CMDWatchBPOfficeReadSettingValues:
            [weakself
                WBOBLEManagerResponseReadSettingValue:model
            ];
            break;
    }
}
```

- 2.88. Read all history or current data from BPM :

	- (void)readAllHistorys;
Definition	Read all history or current data from BPM
CMD	CMDWatchBPOfficeReadHistorys
model	There is measurement data: MicroLifeWatchBPRecord No measurement data: MicroLifeDataModel
	<pre>case CMDWatchBPOfficeReadHistorys: if ([model isKindOfClass:[MicroLifeWatchBPRecord class]]) { [weakself WBOBLEManagerResponseReadAllHistorys :model]; } else { // reply Null ACK } break;</pre>

- 2.89. Read CBP data by index from BPM :

	- (void)readCBPDataWithIndex:(int)index Dformat:(Dformat)dformat;
Definition	Read diagnostic mode history data from BPM
Parameter	<p>index : is CBP memory index</p> <p>dformat : Data format which BPM sent out. It is same as what APP requested.</p> <p>NoCBPRaw : No CBP raw data</p> <p>LowCBPRaw : low resolution CBP data (sampling rate =16Hz)</p> <p>FullCBPRaw : full CBP raw data (sampling rate=256Hz)</p>
CMD	CMDWatchBPOfficeReadCBPData
model	<p>There is measurement data: MicroLifeWatchBPCBPdataAndCalCBP</p> <p>No measurement data: MicroLifeDataModel</p>
	<pre> case CMDWatchBPOfficeReadCBPData: { if ([model isKindOfClass: [MicroLifeWatchBPCBPdataAndCalCBP class]]) { [weakSelf WBOBLEManagerResponseReadCBPData:model el IsNoData:NO]; } else { [weakSelf WBOBLEManagerResponseReadCBPData:nil IsNoData:YES]; } } break; </pre>

2.90. Clear all history data of the BPM :

	- (void)clearAllHistorys;
Definition	clear all history data of the bpm
CMD	CMDWatchBPOfficeClearAllHistorys
model	MicroLifeDataModel
	<pre> case CMDWatchBPOfficeClearAllHistorys: { MicroLifeDataModel *dataModel = model; [weakSelf WBOBLEManagerResponseClearHistory:dataMo del.success.boolValue]; } break; </pre>

2.91. Disconnect the Bluetooth with BPM :

	- (void)disconnect;
Definition	Disconnect the Bluetooth with BPM

Parameter	void(^connectDeviceStateBlock) (id device, CBCentralManager *central, CBPeripheral *peripheral, CBPeripheralState state, NSError * error)
Parameter	void(^cancelAllDevicesConnectionBlock) (void)
	<pre> [self.sdk getConnectDeviceStateBlock:^(id _Nonnull device, CBCentralManager * _Nonnull central, CBPeripheral * _Nonnull peripheral, CBPeripheralState state, NSError * _Nonnull error) { [weakself connectDeviceInfo:device PeripheralState:state]; } CancelAllDevicesConnectionBlock:^([weakself addLogWhitText:@"Cancel All Devices Connection"]; }]]; </pre>

2.92. Read user ID and version data from BPM :

	- (void)readUserAndVersionData;
Definition	Read user ID and version data from BPM: After the device is successfully connected, the user ID and version data will be automatically read.
CMD	CMDWatchBPOfficeReadUserAndVersionData
model	Dictionary : { UserInfo = "<MicroLifeUserInfo: User Information>"; Version = "<MicroLifeDeviceInfo: Device Information>"; }
	<pre> case CMDWatchBPOfficeReadUserAndVersionData: { MicroLifeUserInfo *userInfo = [model valueForKey:@"UserInfo"]; MicroLifeDeviceInfo *version = [model valueForKey:@"Version"]; [weakself WBOBLEManagerResponseReadUserAndVersionData:userInfo VersionData:version]; } break; </pre>

2.93. Write a new user ID to BPM :

	- (void)writeUserID:(NSString *)ID;
Definition	Write a new user ID to BPM
Parameter	ID : User ID.
CMD	CMDWatchBPOfficeWriteUser
model	MicroLifeDataModel
	<pre> case CMDWatchBPOfficeWriteUser: { MicroLifeDataModel *dataModel = model; [weakself WBOBLEManagerResponseWriteUserID:dataModel.success.boolValue]; } break; </pre>

2.94. Read device ID and info from BPM :

	- (void)readDeviceIDAndInfo;
Definition	Read device ID and info from BPM
CMD	CMDWatchBPOfficeReadDeviceInfo
model	MicroLifeDeviceInfo
	<pre> case CMDWatchBPOfficeReadDeviceInfo: [weakSelf WBOBLEManagerResponseReadDeviceIDAndInfo :model]; break; </pre>

2.95. Read device Time from BPM :

	- (void)readDeviceTime;
Definition	Read device Time from BPM
CMD	CMDWatchBPOfficeReadDeviceTime
model	MicroLifeDeviceInfo
	<pre> case CMDWatchBPOfficeReadDeviceTime: [weakSelf WBOBLEManagerResponseReadDeviceTime:mode l]; break; </pre>

2.96. Write device Time to BPM :

	- (void)writeDeviceTime;
Definition	Write device Time to BPM: After the device is successfully connected, the time will be automatically synchronized.
CMD	CMDWatchBPOfficeWriteDeviceTime
model	MicroLifeDataModel
	<pre> case CMDWatchBPOfficeWriteDeviceTime: { MicroLifeDataModel *dataModel = model; [weakSelf WBOBLEManagerResponseWriteDeviceTime:dat aModel.success.boolValue]; } break; </pre>

2.97. Read BPM function setting value from BPM :

	- (void)readFunctionSettingValue;
Definition	Read BPM function setting value from BPM
CMD	CMDWatchBPOfficeReadFunctionSettingValue
model	MicroLifeWatchBPFunctionSettingValues

	<pre> case CMDWatchBPOfficeReadFunctionSettingValue: [weakSelf WBOBLEManagerResponseReadFunctionSetting Value:model]; break; </pre>
--	---

2.98. Read ABPM setting values from BPM :

	- (void)readSettingValues;
Definition	Read ABPM setting values from BPM
CMD	CMDWatchBPOfficeReadSettingValues
model	MicroLifeWatchBPSettingValues
	<pre> case CMDWatchBPOfficeReadSettingValues: [weakSelf WBOBLEManagerResponseReadSettingValue:model]; break; </pre>

2.99. Write ABPM setting values to BPM :

	<p>-</p> <p>(void)writeSettingValuesWithSelectedAUS_HI_infPressure:(HlinfPressure)AUS_HI_infPressure HI_infPressure:(HlinfPressure)HI_infPressure SW_AUTO_hide:(BOOL)SW_AUTO_hide SW_SEL_silent:(BOOL)SW_SEL_silent SW_AUS_Hide:(BOOL)SW_AUS_Hide SW_AVG_no_include_first:(BOOL)SW_AVG_no_include_first SW_CBP:(BOOL)SW_CBP SW_AFib:(BOOL)SW_AFib SW_AMPM:(BOOL)SW_AMPM SW_Kpa:(BOOL)SW_Kpa RestTime:(NSInteger)RestTime IntervalTime:(NSInteger)IntervalTime AutoMeasureNumber:(NSInteger)AutoMeasureNumber;</p>
Definition	Write ABPM setting values to BPM.
Parameter	<p>AUS_HI_infPressure Highest inflation pressure of AUS mode. HI_infPressure Highest inflation pressure of Auto mode. SW_AUTO_hide Set Show readings during rest time in auto mode. true:hide/false:show. SW_SEL_silent Beeper true:enabled/false:disabled SW_AUS_Hide Set Show cuff pressure during deflation in AUS mode. true:hide/false:show. SW_AVG_no_include_first Set Average is include first memory data. true:is/false:is not. SW_CBP Set CBP measurement true:enabled/false:disabled. SW_AFib Set AFib measurement true:enabled/false:disabled. SW_AMPM Set 12/24-hour clock true:12-hour/false:24-hour. SW_Kpa Set Pressure unit: true:Kpa/false:mmHg. RestTime Rest time of auto mode. Start countdown base on rest time before 1st measurement in auto mode.</p>

	IntervalTime Interval time of auto mode. Start countdown base on interval time before 2nd~6th measurement in auto mode. AutoMeasureNumber It's number of measurements in auto mode.
CMD	CMDWatchBPOfficeReadSettingValues
model	MicroLifeDataModel
	<pre>[self.watchBPOffice writeSettingValuesWithSelectedAUS_HI_infPr essure:self.editAUS_HI_infPressure.text .integerValue HI_infPressure:self.edithI_infPressure .text.integerValue SW_AUTO_hide:self.SW_AUTO_hide.on SW_SEL_silent:self.SW_SEL_silent.on SW_AUS_Hide:self.SW_AUS_Hide.on SW_AVG_no_include_first:self .SW_AVG_no_include_first.on SW_CBP:self.SW_CBP.on SW_AFib:self.SW_AFib.on SW_AMPM:self.SW_AMPM.on SW_Kpa:self.SW_Kpa.on RestTime:self.editRestTime.text .integerValue IntervalTime:self.editIntervalTime.text .integerValue AutoMeasureNumber:self .editAutoMeasureNumber.text.integerValue];</pre>
	<pre>case CMDWatchBPOfficeWriteSettingValues: { MicroLifeDataModel *dataModel = model; [weakSelf WBOBLEManagerResponseWriteSettingValues: dataModel.success.boolValue]; } break;</pre>

2.100. Read BT module name from BPM :

	- (void)readBTModuleName;
Definition	Read BT module name from BPM
CMD	CMDWatchBPOfficeReadBTModuleName
model	MicroLifeDeviceInfo
	<pre>case CMDWatchBPOfficeReadBTModuleName: { MicroLifeDeviceInfo *deviceInfo = model; [weakSelf WBOBLEManagerResponseReadBTModuleName:de viceInfo.BTModuleName]; } break;</pre>

2.101. Start remote measurement :

	- (void)startRemoteMeasurementWhithCBPFunction:(Dformat)d format;
Definition	Start remote measurement
Parameter	dformat : Data format which BPM sent out. It is same as what APP requested. NoCBPRaw : No CBP raw data LowCBPRaw : low resolution CBP data (sampling rate =16Hz) FullCBPRaw : full CBP raw data (sampling rate=256Hz)
CMD	CMDWatchBPOfficeStartRemoteMeasurement CMDWatchBPOfficeSendRemoteMeasurementStatusEvery5s econds CMDWatchBPOfficeSendMeasurementResultsForEachMeasu rement
model	Measurement period: MicroLifeWatchBPRemoteMeasurement Measurement results: MicroLifeWatchBPCurrentAndMData
	<pre> case CMDWatchBPOfficeStartRemoteMeasurement: case CMDWatchBPOfficeStopRemoteMeasurement: { MicroLifeWatchBPRemoteMeasurement *rmModel = model; [weakSelf WBOBLEManagerResponseStartRemoteMeasurem ent:rmModel.dformat]; } break; </pre>
	<pre> case CMDWatchBPOfficeSendRemoteMeasurementStatusE very5seconds: { MicroLifeWatchBPRemoteMeasurement *rmModel = model; [weakSelf WBOBLEManagerResponseremoteMeasurementSt atusEvery5secondsWithSTATUS:rmModel .status MeasurementNumber:rmModel .measurementNumber.intValue TotalMeasurementNumber:rmModel .totalMeasurementNumber.intValue Countdown:rmModel.countdown.intValue TotalMeasuretime:rmModel .totalMeasuretime.intValue]; } break; </pre>

	<pre> case CMDWatchBPOfficeSendMeasurementResultsForEachMeasurement: { MicroLifeWatchBPRemoteMeasurement *rmModel = [model valueForKey:@"RMInfo"]; MicroLifeWatchBPCurrentAndMData *mData = [model valueForKey:@"Measurement"]; [weakSelf WBOBLEManagerResponseMeasurementResultsForEachMeasurement:mData HistoryMeasurementNumber:rmModel .historyMeasuremeNumber.intValue CurrentMeasurementTimes:rmModel .currentMeasurementTimes.intValue AverageCalculationWhenMeasurement:rmModel.isAverage.boolValue]; } break; </pre>
--	---

2.102. Stop remote measurement :

	- (void)stopRemoteMeasurement;
Definition	Stop remote measurement
CMD	CMDWatchBPOfficeStopRemoteMeasurement
model	MicroLifeWatchBPRemoteMeasurement
	<pre> case CMDWatchBPOfficeStartRemoteMeasurement: case CMDWatchBPOfficeStopRemoteMeasurement: { MicroLifeWatchBPRemoteMeasurement *rmModel = model; [weakSelf WBOBLEManagerResponseStartRemoteMeasurement:rmModel.dformat]; } break; </pre>

12.manual:

2.103. Independent management mode (multiple devices connected simultaneously):

2.103.1. Singleton Bailuo device:

```
// Device manager
self.watchBP03 = [MicroLifeWatchBP03
    shareWhithAuthorizationkey:SDKkey_WBP];
```

2.103.2. Monitor the Bluetooth status of the mobile phone, device scanning and connection status: When the Bluetooth of the mobile phone is turned on, it will automatically search for surrounding devices, find a matching device, and automatically connect to the device by default. **After the device is connected, if no command is sent for more than 120 seconds, it will automatically disconnect.**

```
[self.watchBP03 getDidUpdateStateBlock:^(CBManagerState state) {
} ScanDeviceBlock:^(id _Nonnull device) {
} CancelScanBlock:^(
} ConnectDeviceStateBlock:^(id _Nonnull device, CBCentralManager
    * _Nonnull central, CBPeripheral * _Nonnull peripheral,
    CBPeripheralState state, NSError * _Nonnull error) {
} CancelAllDevicesConnectionBlock:^(
}];
```

2.103.3. Monitor command response status: send commands according to usage needs, and use CMD to obtain corresponding information.

```
// Device response
[self.watchBP03 getReadDataModelBlock:^(NSInteger CMD, id
    _Nonnull model, NSError * _Nonnull error) {
}];
```

2.103.4. Monitor error response status: When receiving a response, it will automatically disconnect.

```
[self.watchBP03 getValidationErrorBlock:^(NSString * _Nonnull  
    item, NSString * _Nonnull info, NSData * _Nonnull value) {  
  
}];
```

2.104. Unified management mode (single device connection):

2.104.1. Singleton Bluetooth manager:

```
// Bluetooth scanner establishment
self.sdk = [BLESDK shareOne];
```

2.104.2. Singleton Bailuo device:

```
// Device manager
self.watchBPO3 = [MicroLifeWatchBP03
    shareWhithAuthorizationkey:SDKkey_WBP];
```

2.104.3. Monitor the Bluetooth status of mobile phone:

```
// Bluetooth status of mobile phone
[self.sdk getDidUpdateStateBlock:^(CBManagerState state) {
    NSString *log = [NSString
        stringWithFormat:@"DidUpdateState :%ld", (long)state];
    [weakself addLogWhitText:log];
}];
```

2.104.4. Monitor device scan status:

```
[self.sdk getScanDeviceBlock:^(id _Nonnull device) {
    NSString *log = [NSString stringWithFormat:@"ScanDevice
        Name :%@ UUID :%@ mac :%@", [device name], [device
        UUID], [device mac]];
    [weakself addLogWhitText:log];
} CancelScanBlock:^() {
    [weakself addLogWhitText:@"Cancel Scan"];
}];
```

2.104.5. Monitor device connection status:

```
[self.sdk getConnectDeviceStateBlock:^(id _Nonnull device,
    CBCentralManager * _Nonnull central, CBPeripheral * _Nonnull
    peripheral, CBPeripheralState state, NSError * _Nonnull error)
{
    [weakself connectDeviceInfo:device PeripheralState:state];
} CancelAllDevicesConnectionBlock:^(
    [weakself addLogWhitText:@"Cancel All Devices Connection"];
}];
```

2.104.6. Add Bailuo device: When the Bluetooth of the mobile phone is turned on, it will automatically search for surrounding devices, find a matching device, and automatically connect to the device by default. After the device is connected, if no command is sent for more than 120 seconds, it will automatically disconnect.


```
// Bluetooth scanner join device
[self.sdk device:@[self.watchBP03]];
```

- 2.104.7. Monitor command response status: send commands according to usage needs, and use CMD to obtain corresponding information.

```
// Device response
[self.watchBP03 getReadDataModelBlock:^(NSInteger CMD, id
    _Nonnull model, NSError * _Nonnull error) {

}];
```

- 2.104.8. Monitor error response status: When receiving a response, it will automatically disconnect.

```
[self.watchBP03 getValidationErrorBlock:^(NSString * _Nonnull
    item, NSString * _Nonnull info, NSData * _Nonnull value) {

}];
```

2.105. Device automatic scanning mechanism:

- 2.105.1. When CBManagerStatePoweredOn automatically opens scanning [Default: Open]
- 2.105.2. Unified management mode (single device connection), use 3.5. Cancel the automatic scanning mechanism.

```
// stop Auto Scan
[self.sdk autoScan:NO];
```

- 2.105.3. Use independent management mode (multiple devices connected at the same time) and use 4.3. to cancel the automatic scanning mechanism.

```
// stop Auto Scan
[self.watchBP03 autoScan:NO];
```

2.106. Device automatic connection mechanism:

- 2.106.1. When the device is scanned, a matching device is found by default and the device will be automatically connected.
- 2.106.2. Use 4.4. Auto Connect to cancel the automatic connection mechanism.

```
// Auto Connect [Default:Open]
[self.watchBP03 setAutoConnect:NO];
```

- 2.106.3. To monitor the status of the scanning device, you can perform a connection based on the response information of void(^scanDeviceBlock) (id device).

```
// Scanning device response
[self.sdk getScanDeviceBlock:^(id _Nonnull device) {
    NSString *log = [NSString stringWithFormat:@"ScanDevice
        Name : %@ UUID : %@ mac : %@", [device name], [device
        UUID], [device mac]];
    [weakself addLogWhitText:log];
    [weakself.watchBP03 connectDevice];
} CancelScanBlock:^() {
    [weakself addLogWhitText:@"Cancel Scan"];
}];
```

13.Instructions for migrating V1.x to V2.x:

- 2.107. Select Bluetooth unified management mode (single device connection) or independent management mode (multiple devices connected at the same time) according to project requirements, refer to Chapter 6.
- 2.108. Removed the Bluetooth selector of the original V1.x version.

```
// Do any additional setup after loading the view.
self.aWBO3BLEManager = [WBO3BLEManager
    sharedInstanceWhithAuthorizationKey:SDKkey_WBP];
self.aWBO3BLEManager.dataResponseDelegate = self;

self.wbo3BindingDevice = [self.aWBO3BLEManager getBindingDevice];
```

- 2.109. -
(void)WBO3BLEManagerCellPhoneBluetoothDidUpdateState:(MicroLifeBLEState)state, MicroLifeBLEState is changed to CBManagerState, and it is inherited in getDidUpdateStateBlock:^(CBManagerState state).
- 2.110. -
(void)WBO3BLEManagerDidDiscoverBluetoothDeviceMacAddress:(nonnull NSData *)macAddress Name:(nonnull NSString *)name
RSSI:(nonnull NSNumber *)RSSI , has been replaced by
ScanDeviceBlock:^(id _Nonnull device)。
- 2.111. - (void)WBO3BLEManagerDidConnectDevice、
- (void)WBO3BLEManagerDidDisconnectDevice、
- (void)WBO3BLEManagerDidFailToConnectDevice 已被
ConnectDeviceStateBlock:^(id _Nonnull device, CBCentralManager *
_Nonnull central, CBPeripheral * _Nonnull peripheral,
CBPeripheralState state, NSError * _Nonnull error) 取代。

```

// Device
[self.watchBPO3 getDidUpdateStateBlock:^(CBManagerState state) {
    [weakSelf
        WB03BLEManagerCellPhoneBluetoothDidUpdateState:state];
} ScanDeviceBlock:^(id _Nonnull device) {
    // -
    (void)WB03BLEManagerDidDiscoverBluetoothDeviceMacAddress:
    (nonnull NSData *)macAddress Name:(nonnull NSString
    *)name RSSI:(nonnull NSNumber *)RSSI
} CancelScanBlock:^(

} ConnectDeviceStateBlock:^(id _Nonnull device, CBCentralManager
* _Nonnull central, CBPeripheral * _Nonnull peripheral,
CBPeripheralState state, NSError * _Nonnull error) {
    // - (void)WB03BLEManagerDidConnectDevice
    // - (void)WB03BLEManagerDidDisconnectDevice
    // - (void)WB03BLEManagerDidFailToConnectDevice
} CancelAllDevicesConnectionBlock:^(
    // - (void)WB03BLEManagerDidDisconnectDevice
    // - (void)WB03BLEManagerDidFailToConnectDevice
}];

```

- 2.112. - (void)WB03BLEManagerRespondTimeOut has been replaced by
 getValidationErrorBlock:^(NSString * _Nonnull item, NSString *
 _Nonnull info, NSData * _Nonnull value).

```

[self.watchBPO3 getValidationErrorBlock:^(NSString * _Nonnull
    item, NSString * _Nonnull info, NSData * _Nonnull value) {
    // - (void)WB03BLEManagerRespondTimeOut
}];

```

- 2.113. WB03DataResponseDelegate has been
 getReadDataModelBlock:^(NSInteger CMD, id _Nonnull model,
 NSError * _Nonnull error), and the corresponding response data is
 obtained according to CMD.

```
[self.watchBP03 getReadDataModelBlock:^(NSInteger CMD, id
    _Nonnull model, NSError * _Nonnull error) {
    switch (CMD) {
        case CMDWatchBP03ReadHistorys:
        case CMDWatchBP03ReadCBPData:
        case CMDWatchBP03ClearAllHistorys:
        case CMDWatchBP03ReadUserAndVersionData:
        case CMDWatchBP03WriteUser:
        case CMDWatchBP03ReadDeviceInfo:
        case CMDWatchBP03ReadDeviceTime:
        case CMDWatchBP03WriteDeviceTime:
        case CMDWatchBP03ReadSerialNumber:
        case CMDWatchBP03ReadFunctionSettingValue:
        case CMDWatchBP03ReadBTModuleName:
        case CMDWatchBP03ReadSettingValues:
        case CMDWatchBP035SReadSettingValues:
        case CMDWatchBP03WriteSettingValues:
        case CMDWatchBP035SWriteSettingValues:
            break;
        default:
            break;
    }
}
```

- 2.114. - (void)WBO3BLEManagerResponseReadAllHistorys:(nonnull MicroLifeDRecord *)data, MicroLifeDRecord is replaced by MicroLifeWatchBPDRRecord, MicroLifeCurrentAndMData is replaced by MicroLifeWatchBPCurrentAndMData, CMD is equal to CMDWatchBP03ReadHistorys.

```
case CMDWatchBP03ReadHistorys:
    if ([model isKindOfClass:[MicroLifeWatchBPDRRecord
        class]]) {
        [weakSelf
            WBO3BLEManagerResponseReadAllHistorys:model];
    } else {
        // reply Null ACK
    }
    break;
```

- 2.115. - (void)WBO3BLEManagerResponseReadCBPData:(nonnull MicroLifeCBPdataAndCalCBP *)data IsNoData:(BOOL)isNoData, MicroLifeCBPdataAndCalCBP is replaced by MicroLifeWatchBPCBPdataAndCalCBP, CMD is equal to CMDWatchBP03ReadCBPData.

```

case CMDWatchBPO3ReadCBPData:
    if ([model
        isKindOfClass:[MicroLifeWatchBPCBPdataAndCalCBP
            class]]) {
        [weakSelf WBO3BLEManagerResponseReadCBPData:model
            IsNoData:NO];
    } else {
        [weakSelf
            WBO3BLEManagerResponseReadCBPData:nil
            IsNoData:YES];
    }
    break;

```

2.116. -

(void)WBO3BLEManagerResponseReadUserAndVersionData:(nonnull
 MicroLifeUserInfo *)user VersionData:(nonnull MicroLifeDeviceInfo
 *)verData , CMD 等於 CMDWatchBPO3ReadUserAndVersionData。

```

case CMDWatchBPO3ReadUserAndVersionData: {
    MicroLifeUserInfo *userInfo = [model
        valueForKey:@"UserInfo"];
    MicroLifeDeviceInfo *version = [model
        valueForKey:@"Version"];
    [weakSelf
        WBO3BLEManagerResponseReadUserAndVersionData:userInfo
        VersionData:version];
}
break;

```

2.117. - (void)WBO3BLEManagerResponseClearHistory:(BOOL)isSuccess ,
 CMD 等於 CMDWatchBPO3ClearAllHistorys。

```

case CMDWatchBPO3ClearAllHistorys: {
    MicroLifeDataModel *dataModel = model;
    [weakSelf
        WBO3BLEManagerResponseClearHistory:dataModel
        .success.boolValue];
}
break;

```

2.118. - (void)WBO3BLEManagerResponseReadBTModuleName:(nonnull
 NSString *)BTModuleName , CMD 等於
 CMDWatchBPO3ReadBTModuleName。

```

case CMDWatchBP03ReadBTModuleName: {
    MicroLifeDeviceInfo *deviceInfo = model;
    [weakSelf
        WBO3BLEManagerResponseReadBTModuleName:deviceInfo
        .BTModuleName];
}
break;

```

2.119. -

(void)WBO3BLEManagerResponseReadFunctionSettingValue:(nonnull
 MicroLifeFunctionSettingValues *)functionSettingValues ,
 MicroLifeFunctionSettingValues 以
 MicroLifeWatchBPFFunctionSettingValues 取代 , CMD 等於
 CMDWatchBP03ReadFunctionSettingValue。

```

case CMDWatchBP03ReadFunctionSettingValue:
    [weakSelf
        WBO3BLEManagerResponseReadFunctionSettingValue:model];
break;

```

2.120. - (void)WBO3BLEManagerResponseReadDeviceTime:(nonnull
 MicroLifeDeviceInfo *)deviceInfo , CMD 等於
 CMDWatchBP03ReadDeviceTime。

```

case CMDWatchBP03ReadDeviceTime:
    [weakSelf WBO3BLEManagerResponseReadDeviceTime:model];
break;

```

2.121. - (void)WBO3BLEManagerResponseReadSettingValue:(nonnull
 MicroLifeSettingValues *)settingValues ,
 MicroLifeFunctionSettingValues 以 MicroLifeWatchBPSettingValues
 取代 , CMD 等 CMDWatchBP03ReadSettingValues、
 CMDWatchBP035SReadSettingValues。

```

case CMDWatchBP03ReadSettingValues:
case CMDWatchBP035SReadSettingValues:
    [weakSelf
        WBO3BLEManagerResponseReadSettingValue:model];
break;

```

2.122. -

(void)WBO3BLEManagerResponseWriteDeviceTime:(BOOL)isSuccess , CMD 等於 CMDWatchBPO3WriteDeviceTime。

```
case CMDWatchBPO3WriteDeviceTime: {
    MicroLifeDataModel *dataModel = model;
    [weakSelf
        WBO3BLEManagerResponseWriteDeviceTime:dataModel
        .success.boolValue];
}
break;
```

2.123. -

(void)WBO3BLEManagerResponseWriteSettingValues:(BOOL)isSuccess , CMD 等於 CMDWatchBPO3WriteSettingValues、
CMDWatchBPO35SWriteSettingValues。

```
case CMDWatchBPO3WriteSettingValues:
case CMDWatchBPO35SWriteSettingValues: {
    MicroLifeDataModel *dataModel = model;
    [weakSelf
        WBO3BLEManagerResponseWriteSettingValues:dataModel
        .success.boolValue];
}
break;
```

2.124. - (void)WBO3BLEManagerResponseWriteUserID:(BOOL)isSuccess ,
CMD 等於 CMDWatchBPO3WriteUser。

```
case CMDWatchBPO3WriteUser: {
    MicroLifeDataModel *dataModel = model;
    [weakSelf
        WBO3BLEManagerResponseWriteUserID:dataModel
        .success.boolValue];
}
break;
```

2.125. - (void)WBO3BLEManagerResponseReadDeviceIDAndInfo:(nonnull
MicroLifeDeviceInfo *)deviceInfo , CMD 等於
CMDWatchBPO3ReadDeviceInfo。


```
case CMDWatchBP03ReadDeviceInfo:  
    [weakSelf  
        WB03BLEManagerResponseReadDeviceIDAndInfo:model];  
break;
```

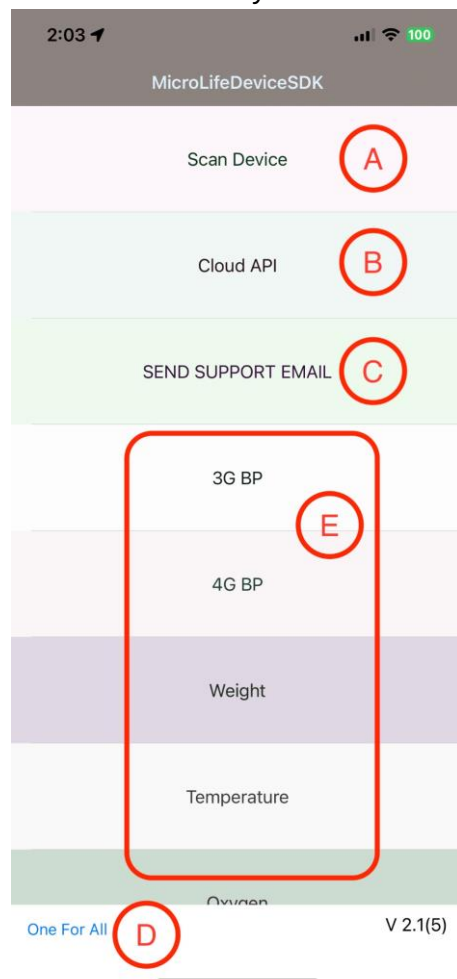
2.126. Read Serial Number, CMD is equal to
CMDWatchBP03ReadSerialNumber.

```
case CMDWatchBP03ReadSerialNumber: {  
    MicroLifeDeviceInfo *deviceInfo = model;  
    NSLog(@"Serial Number:%@",deviceInfo.sn);  
}  
break;
```

14.Dome Code Description:

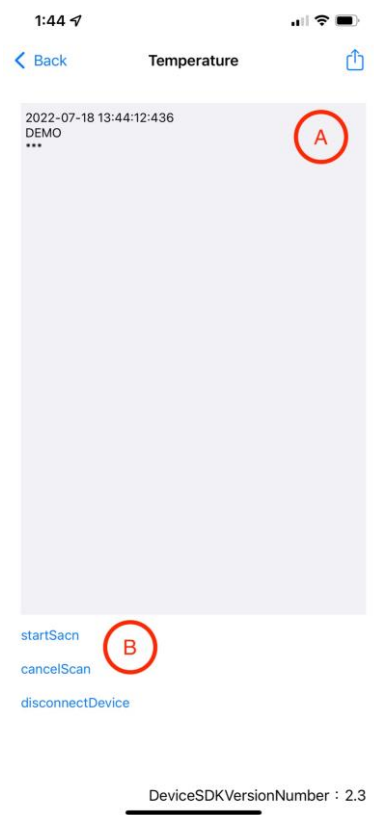
2.127. Dome Code navigation page:

- 2.127.1. A: Scan Device, Bluetooth SDK function display.
- 2.127.2. B: Cloud API, cloud API SDK function display.
- 2.127.3. C: Send local Log to MicroLife.
- 2.127.4. D: One For All, multiple devices are automatically connected and displayed, using independent management mode (multiple devices are connected at the same time).
- 2.127.5. E: Body Temperature SDK, forehead thermometer SDK function display, Blood Pressure SDK, blood pressure machine SDK function display, Oxygen SDK, blood oxygen machine SDK function display, WatchBP O3 II SDK, WatchBP O3 II SDK function display, WatchBP Office II SDK, WatchBP Office II SDK function display, WatchBP HOME SDK, WatchBP Home SDK function display, LTC, WatchBP M1 SDK function display, Blood Sugar SDK, Blood Sugar SDK function display, ESG SDK, ESG SDK function display, Body Composition SDK, body fat machine SDK function demonstration.



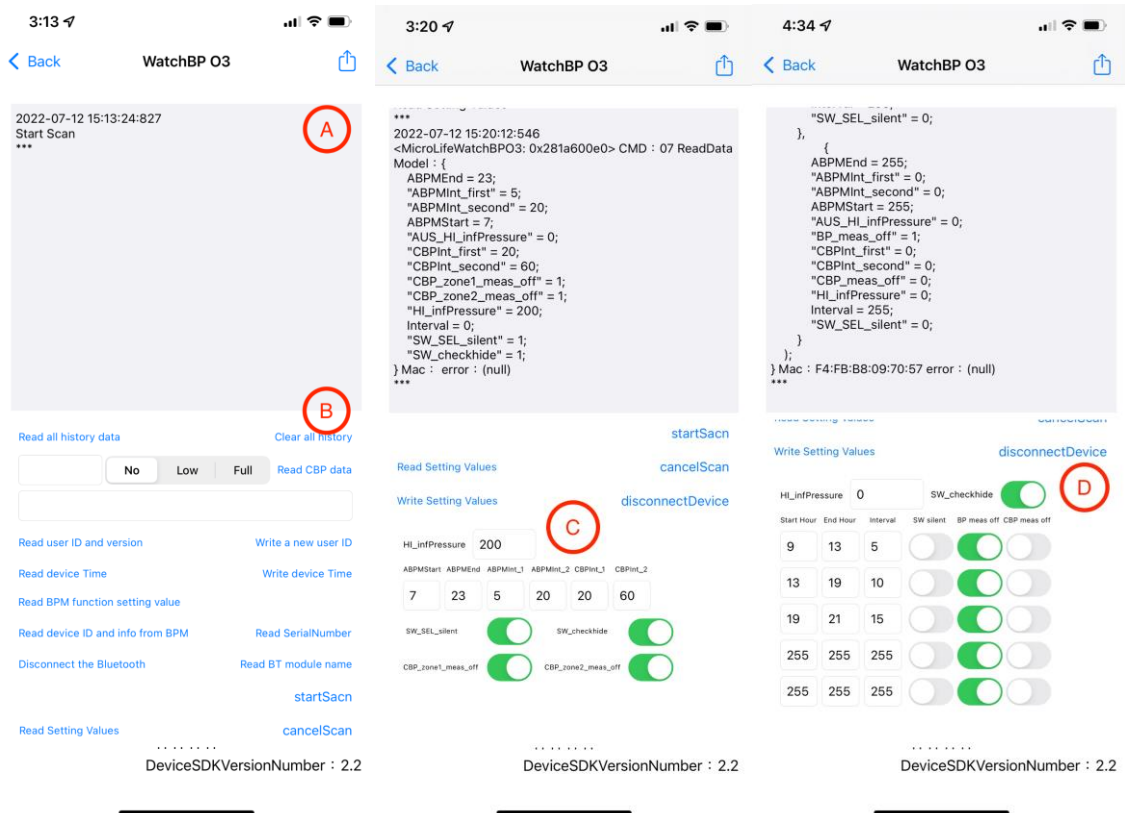
2.128. Temperature SDK interface description:

- 2.128.1. A: Log display area.
- 2.128.2. B: Function command area.
- 2.128.3. For related code, please refer to TEMPViewController.
- 2.128.4. Instructions for use:
 - 2.128.4.1. Entering the display page, a device search will be automatically performed.
 - 2.128.4.2. If a matching device is found, the device will be automatically connected.
 - 2.128.4.3. After the device is connected, if no response is received for more than 120 seconds, it will automatically disconnect.



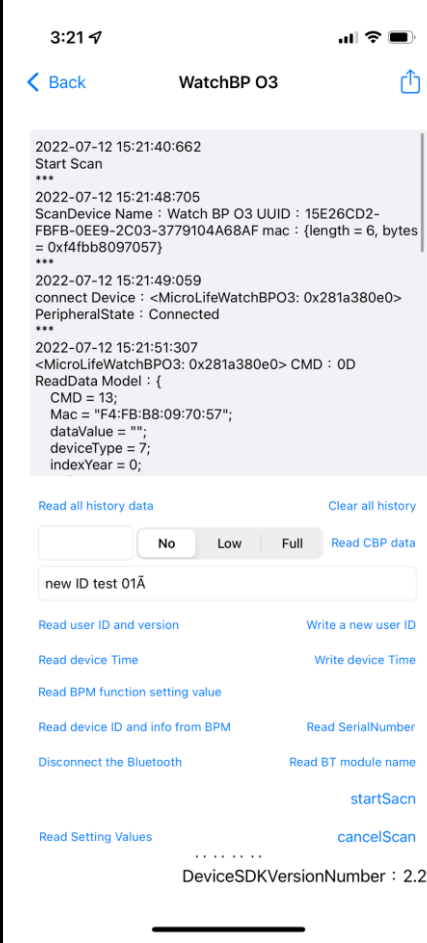
2.129. WatchBP O3 II SDK interface description:

- 2.129.1. A: Log display area.
- 2.129.2. B: Function command area, which can slide up and down.
- 2.129.3. C: Read Setting Values (2 schedule) parameter setting interface.
- 2.129.4. D: Read Setting Values (5 schedule) parameter setting interface.
- 2.129.5. For related code, refer to WBO3ViewController.
- 2.129.6. Instructions for use:
 - 2.129.6.1. Entering the display page, a device search will be automatically performed.
 - 2.129.6.2. If a matching device is found, the device will be automatically connected.
 - 2.129.6.3. After the device is connected, the SDK will automatically send "Write device Time to BPM" and "Read user ID and version data from BPM".
 - 2.129.6.4. After the device is connected, select the function operation in the device function command area.
 - 2.129.6.5. After the device is connected, if no command is sent for more than 120 seconds, it will automatically disconnect.
 - 2.129.6.6. After using "Read Setting Values", the parameter setting interface will be displayed according to WatchBP O3 II (2 schedule / 5 schedule).



15.Instructions

2.130. Search & connect:

 <p>The screenshot shows the 'WatchBP O3' app interface. At the top, there's a status bar with the time 3:21 and signal/battery icons. Below the title 'WatchBP O3', there's a log area with the following text:</p> <pre> 2022-07-12 15:21:40:662 Start Scan *** 2022-07-12 15:21:48:705 ScanDevice Name : Watch BP O3 UUID : 15E26CD2- FBFB-0EE9-2C03-3779104A68AF mac : {length = 6, bytes = 0xf4fbb8097057} *** 2022-07-12 15:21:49:059 connect Device : <MicroLifeWatchBPO3: 0x281a380e0> PeripheralState : Connected *** 2022-07-12 15:21:51:307 <MicroLifeWatchBPO3: 0x281a380e0> CMD : 0D ReadData Model : { CMD = 13; Mac = "F4:FB:B8:09:70:57"; dataValue = ""; deviceType = 7; indexYear = 0; } </pre> <p>Below the log, there are buttons: 'Read all history data', 'Clear all history', and a 'Read CBP data' button with a dropdown menu showing 'No', 'Low', and 'Full'. There's also a text input field with 'new ID test 01Ã'. At the bottom, there are several blue text links: 'Read user ID and version', 'Write a new user ID', 'Read device Time', 'Write device Time', 'Read BPM function setting value', 'Read device ID and info from BPM', 'Read SerialNumber', 'Disconnect the Bluetooth', 'Read BT module name', 'startSacn', 'Read Setting Values', 'cancelScan', and 'DeviceSDKVersionNumber : 2.2'.</p>	<p>1. After entering the display page, the SDK will automatically start searching for and connecting to compatible devices.</p> <p>2. ScanDevice Name : Watch BP O3 UUID : 15E26CD2-FBFB-0EE9-2C03-3779104A68AF mac : {length = 6, bytes = 0xf4fbb8097057}</p> <p>3.connect Device : <MicroLifeWatchBPO3: 0x2825c10a0> PeripheralState : Connected</p>
--	--

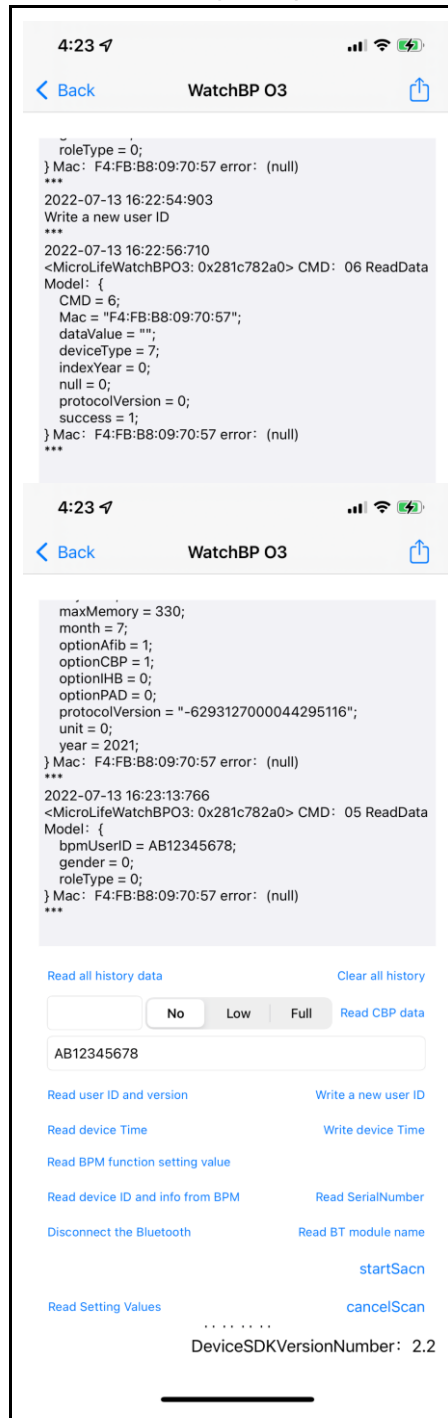
2.131. pair:



1. If the device needs to perform Bluetooth pairing, the Bluetooth pairing dialog window will pop up to ask.

2. After pairing is completed, select the test command.

2.132. Test command (example: Write a new user ID):



1. Select the test command: Write a new user ID.
2. Write a new user ID: AB12345678. The user ID is an Ascii code.
3. <MicroLifeWatchBPO3: 0x281c782a0> CMD : 06
ReadData Model : {
 CMD = 6;
 Mac = "F4:FB:B8:09:70:57";
 dataValue = "";
 deviceType = 7;
 indexYear = 0;
 null = 0;
 protocolVersion = 0;
 success = 1;
} Mac : F4:FB:B8:09:70:57
error : (null)
: Write status response.
4. You can use "Read user ID and version data" to check the newly written user ID.

2.133. Test instructions (example: Read user ID and version data):

4:08

< Back

WatchBP O3

```

2022-07-13 16:07:43:919
<MicroLifeWatchBP03: 0x2825c10a0> CMD: 05 ReadData
Dictionary: {
  UserInfo = "<MicroLifeUserInfo: 0x282ddd400>";
  Version = "<MicroLifeDeviceInfo: 0x121620310>";
} error: (null)
***
2022-07-13 16:07:43:927
<MicroLifeWatchBP03: 0x2825c10a0> CMD: 05 ReadData
Model: {
  FWName = RE1;
  batteryVoltage = 4;
  day = 23;
  maxMemory = 330;
  month = 7;
  optionAfib = 1;
  optionCBP = 1;
  optionHB = 0;
  optionPAD = 0;
  protocolVersion = "-5117805771225310652";
  unit = 0;
} Mac: F4:FB:B8:09:70:57 error: (null)
***
2022-07-13 16:07:43:927
<MicroLifeWatchBP03: 0x2825c10a0> CMD: 05 ReadData
Model: {
  FWName = RE1;
  batteryVoltage = 4;
  day = 23;
  maxMemory = 330;
  month = 7;
  optionAfib = 1;
  optionCBP = 1;
  optionHB = 0;
  optionPAD = 0;
  protocolVersion = "-5117805771225310652";
  unit = 0;
  year = 2021;
} Mac: F4:FB:B8:09:70:57 error: (null)
***
2022-07-13 16:07:43:934
<MicroLifeWatchBP03: 0x2825c10a0> CMD: 05 ReadData
Model: {
  bpmUserID = "new ID test 01\U00c3";
  gender = 0;
  roleType = 0;
} Mac: F4:FB:B8:09:70:57 error: (null)
***

```

Read all history data

Clear all history

Read CBP data

Read user ID and version

Write a new user ID

Read device Time

Write device Time

Read BPM function setting value

Read SerialNumber

Read device ID and info from BPM

Read BT module name

Disconnect the Bluetooth

startSacn

Read Setting Values

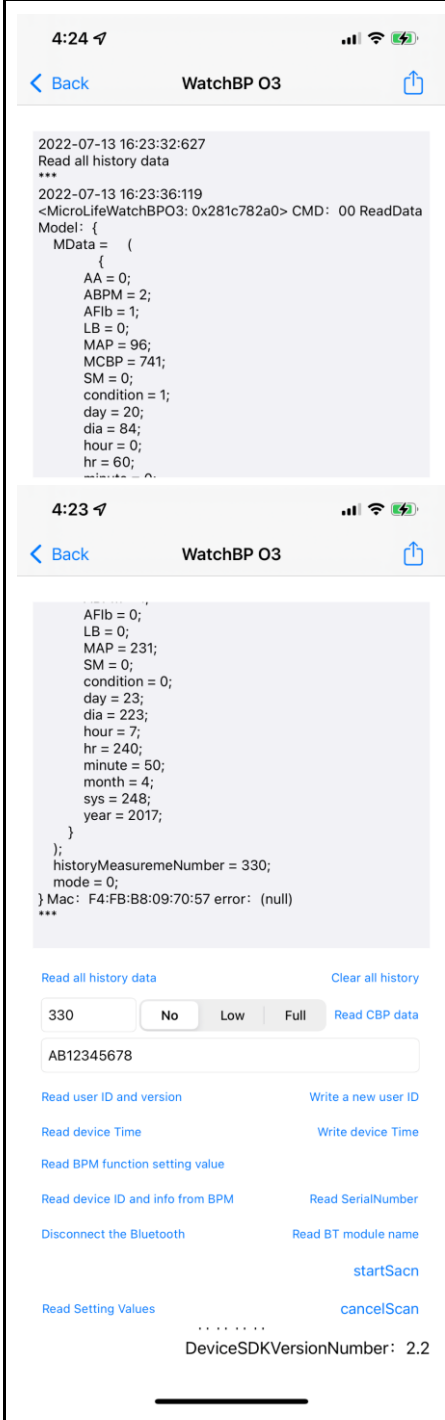
cancelScan

.....

DeviceSDKVersionNumber: 2.2

1. Select the test command: Read user ID and version data.
2. Response:
 <MicroLifeWatchBP03: 0x2825c10a0> CMD: 05 ReadData Dictionary: {
 UserInfo =
 "<MicroLifeUserInfo: 0x282ddd400>";
 Version =
 "<MicroLifeDeviceInfo: 0x121620310>";
 } error : (null)
3. MicroLifeUserInfo: User information
4. MicroLifeDeviceInfo: device information

2.134. Test command (example: Read all history data):



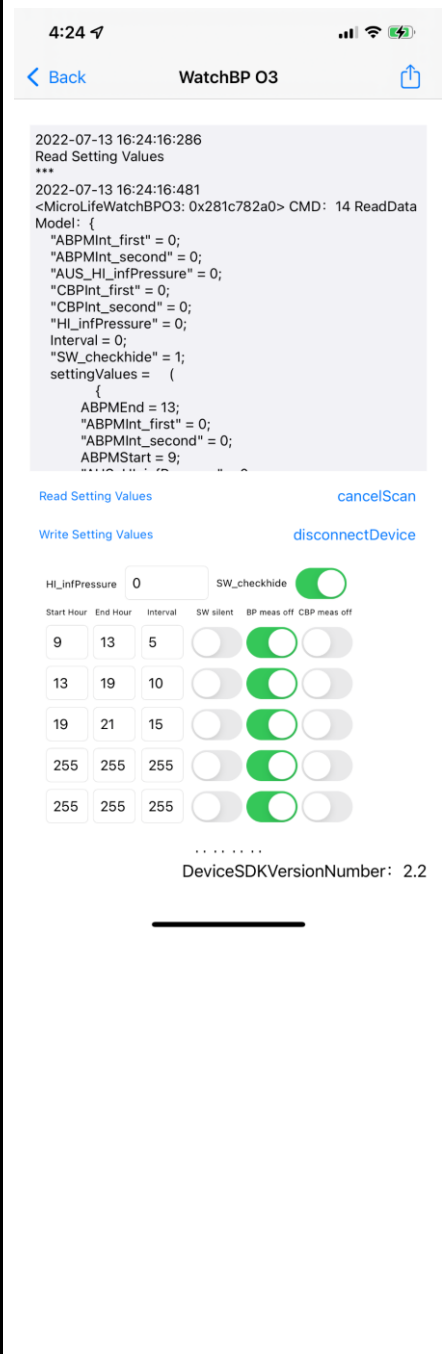
1. Select the test command:
Read all history data.
2. <MicroLifeWatchBP03:
0x281c782a0> CMD : 00

```

ReadData Model : {
    MData = (
        {
            AA = 0;
            ABPM = 2;
            AFib = 1;
            LB = 0;
            MAP = 96;
            MCBP = 741;
            SM = 0;
            condition = 1;
            day = 20;
            dia = 84;
            hour = 0;
            hr = 60;
            minute = 0;
            month = 4;
            sys = 120;
            year = 2017;
        },
        {...},
        ...,
        {...}
    );
    historyMeasuremeNumber = 330;
    mode = 0;
}
Mac: F4:FB:B8:09:70:57 error: (null)

```

2.135. Test command (example: Read Setting values):



1. Select the test command:
Read Setting values.
2. <MicroLifeWatchBPO3:
0x281c782a0> CMD : 14

ReadData Model : {

```

"ABPMInt_first" = 0;
"ABPMInt_second" = 0;
"AUS_HI_infPressure" = 0;
"CBPInt_first" = 0;
"CBPInt_second" = 0;
"HI_infPressure" = 0;
Interval = 0;
"SW_checkhide" = 1;
settingValues = (
{
ABPMEnd = 13;
"ABPMInt_first" = 0;
"ABPMInt_second" = 0;
ABPMStart = 9;
"AUS_HI_infPressure" =
0;
"BP_meas_off" = 1;
"CBPInt_first" = 0;
"CBPInt_second" = 0;
"CBP_meas_off" = 0;
"HI_infPressure" = 0;
Interval = 5;
"SW_SEL_silent" = 0;
},
{...},
{...},
{...},
{...}
);
} Mac : F4:FB:B8:09:70:57
error : (null)

```

16.Update record :

[iOS SDK Update History](#)