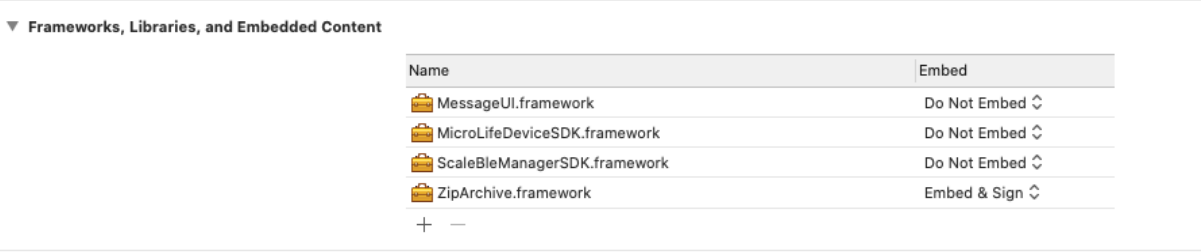
MicroLifeDeviceSDK iOS

Chapter

1. Environment Configuration Instructions:
    1.1. SDK Minimum Supported Version: iOS 13
    1.2. Drag MicroLifeDeviceSDK.framework into in the project.
    1.3. Add MicroLifeDeviceSDK.framework to TARGETS / General / Frameworks, Libraries, and Embedded Content.
    1.4. To get the local Log, you need to add ZipArchive.framework in MicroLifeDeviceSDK/framework to frameworks, Libraries, and Embedded Content, and set "Embed & Sign".
    1.5. To use the body fat machine, you need to add the BelterScaleInfo.framework in MicroLifeDeviceSDK/framework to frameworks, Libraries, and Embedded Content, and set "Do Not Embed".



    1.6. Add -all_load in TARGETS / Build Settings / Other Linkeder Flags.
    1.7. Where you need to use it, just import the header file.

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

2. SDK tool description:
    2.1. Log Management: Local Log recording tool

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

    2.2. Bluetooth Core:
        2.2.1. BLESDK: Bluetooth bottom manager, responsible for all Bluetooth general operations.
        2.2.2. BLEDeviceInfo: Bluetooth device tool, Bluetooth scan response basic object.
        2.2.3. MicroLifeDevice: MicroLife Device Tool. Responsible for the transmission and parsing of all kinds of equipment instructions of Paragon.
        2.2.4. MicroLifeDataModel: MicroLife's general data object.
        2.2.5. MicroLifeDeviceInfo: MicroLife device object.
        2.2.6. MicroLifeUserInfo: MicroLife 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>
```

    2.3. Temperature
        2.3.1. MicroLifeTemperature: MicroLifeTemperature device tool.
        2.3.2. MicroLifeTemperatureMeasureData: Measure Data

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

3. SDK Bluetooth interface description:

3.1. Singleton:

	+ (instancetype)shareOne;
Definition	Singleton

3.2. instantiation:

	+ (instancetype)share;
Definition	instantiation

3.3. Show Log:

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

3.4. Set MicroLife Device:

	- (void)device:(NSArray *)bluetooth;
Definition	device. When the Bluetooth of the mobile phone is turned on, it will automatically scan the surrounding devices and find the devices that match. , the device is automatically connected by default.
Parameter	bluetooth: MicroLifeDevice Class
	`[self.sdk device:@[self.bp3g,self.bp4g,self.temperature,self.oxygen,self.weight,self.bloodSugar]];`

3.5. Set Auto Scan:

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

3.6. Set Scan stop Time:

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

3.7. Set RSSI:

	- (void)rssi:(NSInteger)rssi;
DefinitionSet	scan RSSI strength
Parameter	rssi: Scan RSSI strength

3.8. Start Scan:

	- (void)startScan;
Definition	Start Scan

3.9. Cancel Scan:

	- (void)cancelScan;
Definition	Cancel Scan

3.10. Cancel All Connect:

	- (void)cancelAllConnect;
Definition	Cancel All Connect

3.11. Get the currently connected devices:

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

3.12. When CentralManager state changed:

	- (void)getDidUpdateStateBlock:(didUpdateStateBlock)block;
DefinitionMonitor	Bluetooth state of the mobile
phoneParameter	void(^didUpdateStateBlock) (CBManagerState state)

3.13. Scan Device:

	- (void)getScanDeviceBlock:(scanDeviceBlock)scanDeviceBlock CancelScanBlock:(cancelScanBlock)cancelScanBlock;
DefinitionMonitor	scan device-like State
Parameter	void(^scanDeviceBlock) (id device)
Parameter	void(^cancelScanBlock) (void)

3.14. Connect Device State:

	- (void)getConnectDeviceStateBlock:(connectDeviceStateBlock) connectDeviceStateBlock CancelAllDevicesConnectionBlock:(cancelAllDevicesConnectionBlock)cancelAllDevicesConnectionBlock;
DefinitionMonitor	connected device
stateParameter	void(^connectDeviceStateBlock) (id device, CBCentralManager *central, CBPeripheral *peripheral, CBPeripheralState state, NSError * error)
Parameter	void(^cancelAllDevicesConnectionBlock) (void)

4. MicroLifeDevice interface description:

4.1. Singleton:

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

4.2. Add Device Model:

	- (void)addDeviceModel:(NSArray *)models;
Definition	Search device custom model
Parameter	models: device name which can be included single or multiple
	`[self.watchBPO3 addDeviceModel:@[@"Watch BP O3"]];`

4.3. Set Auto Scan:

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

4.4. Auto Connect [Default: Open]:

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

4.5. ReConnect Time [Default: 0]:

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

4.6. Set Scan stop Time:

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

4.7. Set RSSI:

	- (void) rssi:(NSInteger)rssi;
Definition	Set the scan RSSI strength, use
Parameter	rssi: Scan RSSI strength

4.8. Start Scan:

	- (void)startScan;
Definition	Start Scan, Bluetooth independent management mode use

4.9. Cancel Scan:

	- (void)cancelScan;

Definition	Cancel Scan, Bluetooth independent management mode uses

4.10. Connect:

	- (void)connectDevice;
Definition	Connect

4.11. Disconnect:

	- (void)disconnectDevice;
Definition	Disconnect

4.12. Set Auto Reconnect:

	(void)setAutoReconnect
;	-Default start

4.13. Cancel Auto Reconnect:

	- (void)cancelAutoReconnect;
DefinitionCancel	automatic reconnection

4.14. Device Connected:

	- (void)deviceConnectedBlock:(deviceConnectedBlock)block;
DefinitionMonitor	device has registered all default services
Parameter	void(^deviceConnectedBlock) (voi d);

4.15. Device independent bluetooth:

	- (void)getDidUpdateStateBlock:(didUpdateStateBlock)didUpdateStateBlock ScanDeviceBlock:(scanDeviceBlock)scanDeviceBlock CancelScanBlock:(cancelScanBlock)cancelScanBlock ConnectDeviceStateBlock:(connectDeviceStateBlock)connectDeviceStateBlock CancelAllDevicesConnectionBlock:(cancelAllDevicesConnectionBlock)
cancelAllDevicesConnectionBlock	; Mobile phone Bluetooth status, device scan 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)

4.16. Read RSSI:

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

4.17. Read Data Value:

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

4.18. Read Data Model:

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

4.19. Read Validation item Error:

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

5. Temperature interface indicates that
   5.1. Temperature is active, and the SDK only provides parsing functions.
   5.2. 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;
    }
}];
```

5.3. Read Device Info:

Definition	RRead Device Info
CMD	CMDTemperatureReadDeviceInfo
model	MicroLifeDeviceInfo

5.4. Read Measure Data:

Definition	Read Measure Data
CMD	CMDTemperatureReadMeasureData
model	MicroLifeTemperatureMeasureData

6. Usage process:

6.1. Independent management mode (multiple devices connected at the same time):

6.1.1. single Instantiate the Parallax device:

```
self.temperature = [MicroLifeTemperature
    shareWhithAuthorizationkey:SDKkey_BT];
```

6.1.2. monitor the Bluetooth status of the mobile phone, device scanning and connection status: When the mobile phone Bluetooth is turned on, it will automatically scan the 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.temperature getDidUpdateStateBlock:^(CBManagerState state) {

} ScanDeviceBlock:^(id  _Nonnull device) {

} CancelScanBlock:^{

} ConnectDeviceStateBlock:^(id  _Nonnull device, CBCentralManager
    * _Nonnull central, CBPeripheral * _Nonnull peripheral,
    CBPeripheralState state, NSError * _Nonnull error) {

} CancelAllDevicesConnectionBlock:^{

}];
```

6.1.3. Monitoring command response status: Send commands according to usage needs, and obtain corresponding information through 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;
    }
}];
```

6.1.4. Monitor error response status: When a response is received, it will automatically disconnect.

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

}];
```

6.2. Unified management mode (single device connection):

6.2.1. single instance Bluetooth manager:

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

6.2.2. single instance Parallax device:

```
self.temperature = [MicroLifeTemperature
    shareWhithAuthorizationkey:SDKkey_BT];
```

6.2.3. monitor mobile phone Bluetooth status:

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

6.2.4. 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"];
}];
```

6.2.5. 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"];
}];
```

6.2.6. join Parallax device: when mobile phone Bluetooth After it is turned on, it will automatically scan the 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.sdk device:@[self.temperature]];
```

6.2.7. Monitoring command response status: Send commands according to usage needs, and obtain corresponding information through 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;
    }
}];
```

6.2.8. Monitor error response status: When a response is received, it will automatically disconnect.

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

}];
```

6.3. Device automatic scanning mechanism:

6.3.1. When CBManagerStatePoweredOn automatically opens scanning [Default: Open]

6.3.2. unified management mode (single device connection), use 3.5. Cancel the automatic scanning mechanism.

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

6.3.3. Use independent management mode (multiple devices are connected at the same time), use 4.3. Cancel the automatic scanning mechanism.

```
[self.temperature autoScan:NO];
```

6.4. Device automatic connection mechanism:

6.4.1. When a device is scanned, it will automatically connect to the device if it finds a matching device by default.

6.4.2. Use 4.4. Auto Connect to cancel the automatic connection mechanism.

```
[self.temperature setAutoConnect:NO];
```

6.4.3. To monitor the scan device status, you can perform connection according to the response information of void(^scanDeviceBlock) (id device).

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

7. V1.x Migration V2.x Instructions:
   7.1. According to the project requirements, choose the Bluetooth unified management mode (single device connection) or independent management mode (multiple devices connected at the same time), refer to Chapter 6.
   7.2. Remove the bluetooth selector of the original V1.x version.

```
// Do any additional setup after loading the view.
self.aTemperatureBLEManager = [TemperatureBLEManager
    shareInstanceWhithAuthorizationkey:SDKkey_BT];
self.aTemperatureBLEManager.dataResponseDelegate = self;
```

   7.3. -(void)TemperatureBLEManagerCellPhoneBluetoothDidUpdateState:(MicroLifeBLEState)state, change MicroLifeBLEState to CBManagerState, and inherit it in getDidUpdateStateBlock:^(CBManagerState state).
   7.4. -(void)TemperatureBLEManagerDidDiscoverBluetoothDeviceMacAddress:(nonnull NSData *)macAddress Name:(nonnull NSString *)name RSSI:(nonnull NSNumber *)RSSI , replaced by ScanDeviceBlock:^(id _Nonnull device).
   7.5. - (void)TemperatureBLEManagerDidConnectDevice,
   - (void)TemperatureBLEManagerDidDisconnectDevice,
   - (void)TemperatureBLEManagerDidFailToConnectDevice have been replaced by ConnectDeviceStateBlock:^(id _Nonnull device, CBCentralManager * _Nonnull central, CBPeripheral * _Nonnull peripheral, CBPeripheralState state, NSError * _Nonnull error).

```
[self.temperature getDidUpdateStateBlock:^(CBManagerState state) {

} ScanDeviceBlock:^(id  _Nonnull device) {

} CancelScanBlock:^{

} ConnectDeviceStateBlock:^(id  _Nonnull device, CBCentralManager
    * _Nonnull central, CBPeripheral * _Nonnull peripheral,
    CBPeripheralState state, NSError * _Nonnull error) {

} CancelAllDevicesConnectionBlock:^{

}];
```

   7.6. - (voidTemperatureBLEManagerRespondTimeOut has been replaced by getValidationErrorBlock:^(NSString * _Nonnull item, NSString * _Nonnull info, NSData * _Nonnull value) .

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

}];
```

7.7. TemperatureDataResponseDelegate has been replaced by getReadDataModelBlock:^(NSInteger CMD, id _Nonnull model, NSError * _Nonnull error), corresponding to CMD Response data.

```
[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;
    }
}];
```

7.8. - (void)TemperatureBLEManagerResponseDeviceInfo:(NSString *)macAddress workMode:(int)workMode batteryVoltage:(float)batteryVoltage, CMD is equal to CMDTemperatureReadDeviceInfo.

```
case CMDTemperatureReadDeviceInfo: {
    MicroLifeDeviceInfo *deviceInfo = model;
    [weakself
        TemperatureBLEManagerResponseDeviceInfo:deviceInfo
        .macAddress workMode:deviceInfo.workMode.intValue
        batteryVoltage:deviceInfo.batteryVoltage
        .floatValue];
}
    break;
```

7.9. - (void)TemperatureBLEManagerResponseUploadMeasureData:(MicroLifeThermoMeasureData *)data; MicroLifeThermoMeasureData is replaced by MicroLifeTemperatureMeasureData, CMD is equal to CMDTemperatureReadMeasureData.

```
case CMDTemperatureReadMeasureData:
    [weakself
        TemperatureBLEManagerResponseUploadMeasureData:mod
        el];
    break;
```

8. Dome Code Description:

8.1. Dome Code navigation page:

8.1.1. A: Body Temperature SDK, function display of forehead temperature gun SDK.

8.1.2. B: Blood Pressure SDK, function display of blood pressure machine SDK.

8.1.3. C: Oxygen SDK, function display of blood oxygen machine SDK.

8.1.4. D : Other SDK, go to the second navigation page.

8.1.5. E: Scan Device, Bluetooth SDK function display.

8.1.6. F: Cloud API, Cloud API SDK function display.

8.1.7. G: Send local Log to MicroLife.

8.1.8. H: One For All, automatic multi-device Connection display, independent management mode (multiple devices connected at the same time).

8.1.9. I: WatchBP O3 II SDK, WatchBP O3 II SDK function display.

8.1.10. J: WatchBP Office II SDK, WatchBP Office II SDK function display.

8.1.11. K: WatchBP HOME SDK, WatchBP Home SDK function display.

8.1.12. L: LTC, WatchBP M1 SDK function display.

8.1.13. M: Blood Sugar SDK, Blood Sugar SDK function display.

8.1.14. N: ESG SDK, ESG SDK function display.

8.1.15. O: Body Composition SDK, body fat machine SDK function display.

MicroLifeDeviceSDK iOS

8.2. Temperature SDK interface description:

8.2.1. A: Log display area.

8.2.2. B: Function command area.

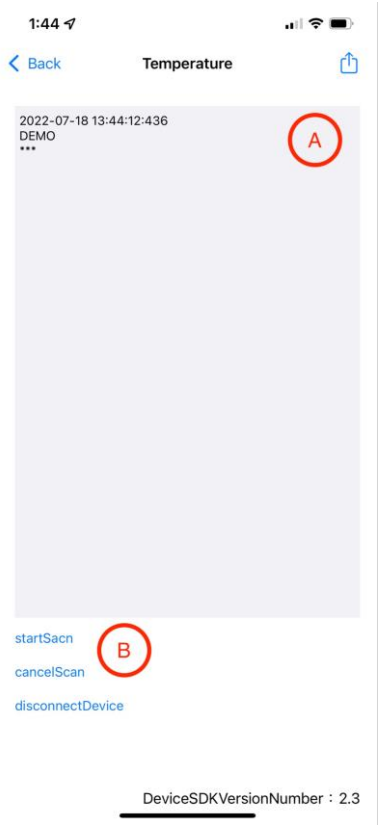
8.2.3. For related code, refer to TEMPViewController.

8.2.4. Instructions for use:

8.2.4.1. Entering the display page will automatically perform a device search.

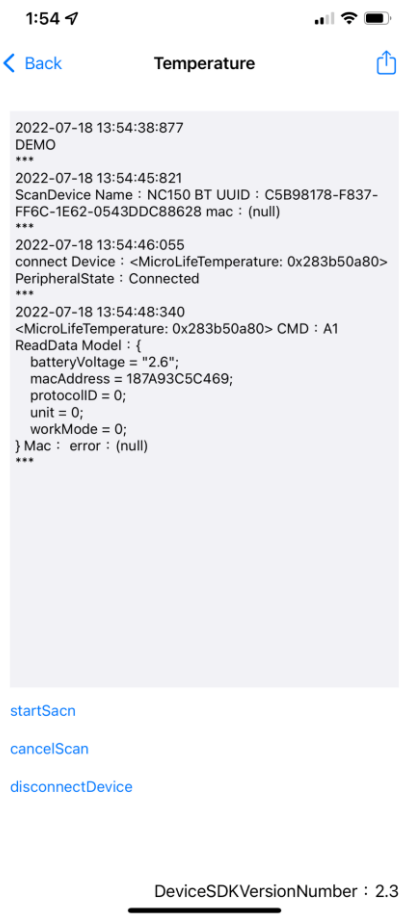
8.2.4.2. If a matching device is found, the device will be automatically connected.

8.2.4.3. After the device is connected, if no response is received for more than 120 seconds, it will automatically disconnect.



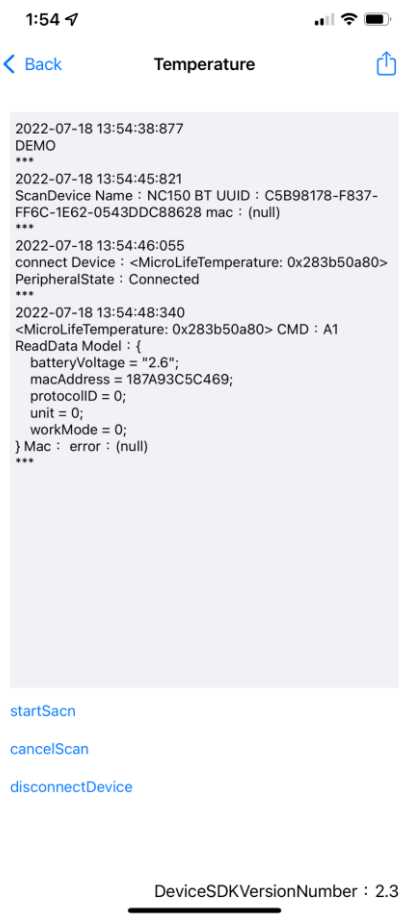
9. Operation Instructions

9.1. Search & Connect:



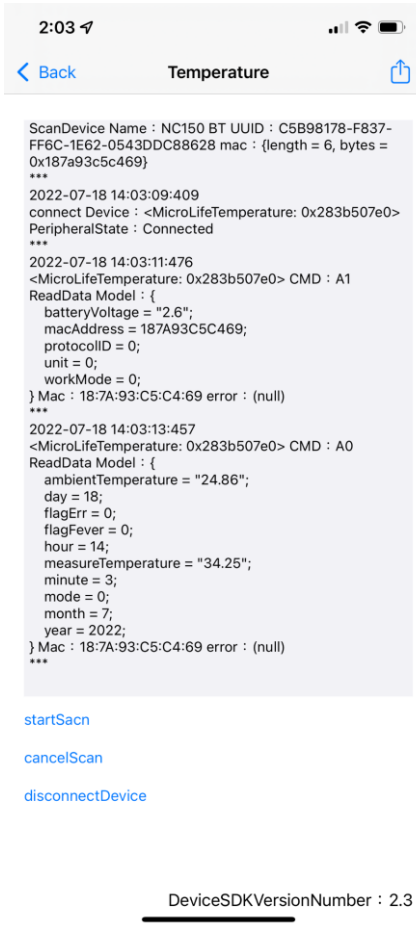
1. After entering the display page, the SDK will automatically start searching and connecting to the matching device.
2. 2022-07-18 14:01:35:792 ScanDevice Name: NC150 BT UUID: C5B98178-F837-FF6C-1E62-0543DDC88628 mac: {length = 6, bytes = 0x187a93c5c469}
3. connect Device: <MicroLifeTemperature: 0x283b507e > PeripheralState: Connected

9.2. response command (eg: CMDTemperatureReadDeviceInfo):



1.<MicroLifeTemperature: 0x283b507e0> CMD: A1 ReadData Model: {
batteryVoltage = "2.6";
macAddress = 187A93C5C469;
protocolID = 0;
unit = 0;
workMode = 0;
} Mac : 18:7A:93:C5:C4:69 error: (null)

9.3. response command (eg: CMDTemperatureReadMeasureData):



<MicroLifeTemperature: 0x283b507e0> CMD: A0
ReadData Model: {
ambientTemperature = "24.86";
day = 18;
flagErr = 0 ;
flagFever = 0;
hour = 14;
measureTemperature = "34.25";
minute = 3;
mode = 0;
month = 7;
year = 2022;
} Mac: 18:7A:93:C5:C4:69 error: (null)

10. Update record:
    - 10.1. V2.x vs V1.x Difference description:
        - 10.1.1. All requests and responses are changed from delegate mode to block mode.
        - 10.1.2. Separate general-purpose tools and Perrier-specific tools.
        - 10.1.3. Data object that separates BPM and WatchBP.
        - 10.1.4. Added Bluetooth unified management mode (single device connection) and independent management mode (multiple devices connected at the same time).
    - 10.2. V2.0: Support device series BPM, BT, Oxygen
    - 10.3. V2.1: Support device series WatchBP home, WatchBP O3(2 schedule), WatchBP office
    - 10.4. V2.2: Support device series WatchBP O3(5 schedule)
    - 10.5. V2.3:
        - 10.5.1. Correction: - (void)startSacn; >> - (void)startScan;.
        - 10.5.2. Added: - (void)autoScan:(BOOL)autoScan;, When CBManagerStatePoweredOn automatically opens scanning [Default: Open]
        - 10.5.3. Fixed: Some commands that need to determine the device version are changed from - (void) to - (BOOL)

MicroLifeDeviceSDK iOS