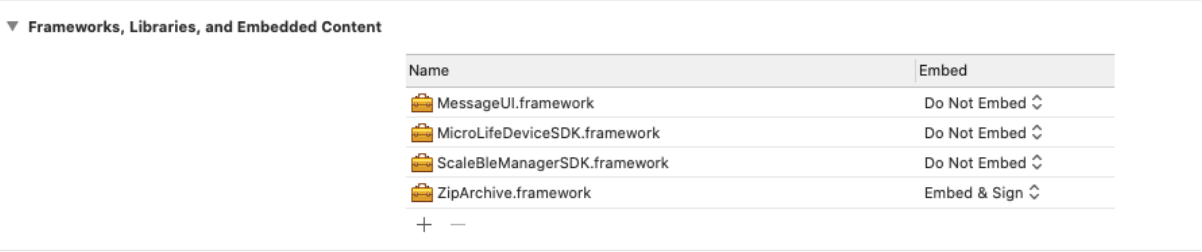
MicroLifeDeviceSDK iOS

Chapter

1. Environment Configuration:
   - 1.1. SDK Minimum Supported Version: iOS 13
   - 1.2. Drag MicroLifeDeviceSDK.framework into 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. MicroLifeDeviceSDK:
    2.1. Log Management: Local log recording tool

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

    2.2. Bluetooth Core:
        2.2.1. BLESDK: Bluetooth core 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.
        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. Blood Pressure:
        2.3.1. MicroLife3GBP: 3G BP device tool.
        2.3.2. MicroLife4GBP: 4G BP device tool.
        2.3.3. MicroLifeBloodPressureDRecord: DRecord.
        2.3.4. MicroLifeBloodPressureCurrentAndMData: Current, MData.

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

3. Bluetooth Interface:

3.1. Singleton:

	+ (instancetype)shareOne;
Definition	Singleton

3.2. instantiation:

	+ (instancetype)share;
Definition	instantiation

3.3. Show Log:

	- (void)showLog:(BOOL)showLog;
Definition	shows Log
Parameter	showLog: Whether to display the Log

3.4. Set MicroLife Device:

	- (void)device:(NSArray *)bluetooth;
Definition	to set the MicroLife device, when the mobile phone's Bluetooth is turned on, it will automatically scan the surrounding devices, find the matching device, the default is set to automatically connect the device.
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	Bluetooth state of Scan
Parameter	void(^scanDeviceBlock) (id device)
Parameter	void(^cancelScanBlock) (void)

3.14. Connect Device State:

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

4. MicroLife Device Interface:

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 model
Parameter	models: device name which can be included single or multiple
	`[self.bp4g addDeviceModel:@[@"6024N"]];`

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) (void);

4.15. Device independent bluetooth:

	- (void)getDidUpdateStateBlock:(didUpdateStateBlock)didUpdateStateBlock ScanDeviceBlock:(scanDeviceBlock)scanDeviceBlock CancelScanBlock:(cancelScanBlock)cancelScanBlock ConnectDeviceStateBlock:(connectDeviceStateBlock)connectDeviceStateBlock CancelAllDevicesConnectionBlock:(cancelAllDevicesConnectionBlock)
cancelAllDevicesConnectionBlock	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 raw data).
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.
Parameter	void(^validationErrorBlock) (NSString *item, NSString *info, NSData * value);

5. Blood Pressure Interface

5.1. Construct the corresponding management handler, 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;
    }
}];
```

5.2. Read all history or current data from BPM:

	- (void)readAllHistorys;
Definition	Read all history or current data from BPM
CMD	CMD3GReadHistorys CMD4GReadHistorys
model	MicroLifeBloodPressureDRecord

	`case CMD3GReadHistorys:`
	`[weakself BPMBLEManagerResponseReadHistory:model];`
	`break;`

	case CMD4GReadHistorys: [weakself BPMBLEManagerResponseReadHistory:model]; break;

5.3. Clear all history data of the BPM:

	- (void)clearAllHistorys;
Definition	clear all history data of the bpm
CMD	CMD3GClearAllHistorys CMD4GClearAllHistorys
model	MicroLifeDataModel

	case CMD3GClearAllHistorys: {
	MicroLifeDataModel *data = model;
	[weakself
	BPMBLEManagerResponseClearHistory:data.success
	.boolValue];
	}
	break;
	case CMD4GClearAllHistorys: {
	MicroLifeDataModel *data = model;
	[weakself
	BPMBLEManagerResponseClearHistory:data.success
	.boolValue];
	}
	break;

5.4. 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)

	[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"];
	}];

5.5. 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>";
	}

```
case CMD3GReadUserAndVersionData: {
    MicroLifeUserInfo *userInfo = [model
        valueForKey:@"UserInfo"];
    MicroLifeDeviceInfo *version = [model
        valueForKey:@"Version"];
    [weakself
        BPMBLEManagerResponseReadUserAndVersionData:userIn
        fo VersionData:version];
}
    break;
```

```
case CMD4GReadUserAndVersionData: {
    MicroLifeUserInfo *userInfo = [model
        valueForKey:@"UserInfo"];
    MicroLifeDeviceInfo *version = [model
        valueForKey:@"Version"];
    [weakself
        BPMBLEManagerResponseReadUserAndVersionData:userIn
        fo VersionData:version];
}
    break;
```

5.6. 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

```
case CMD3GWriteUser: {
    MicroLifeDataModel *data = model;
    [weakself
        BPMBLEManagerResponseWriteUserID:data.success
        .boolValue];
}
    break;
```

```
case CMD4GWriteUser: {
    MicroLifeDataModel *data = model;
    [weakself
        BPMBLEManagerResponseWriteUserID:data.success
        .boolValue];
}
    break;
```

5.7. Read device ID and info from BPM:

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

```
case CMD3GReadDeviceInfo:
    [weakself BPMBLEManagerResponseReadDeviceInfo:model];
    break;
```

```
case CMD4GReadDeviceInfo:
    [weakself BPMBLEManagerResponseReadDeviceInfo:model];
```

5.8. [4G Dedicated] Read device Time from BPM:

	- (void)readDeviceTime;
Definition	Read device Time from BPM
CMD	CMD4GReadDeviceTime
model	MicroLifeDeviceInfo

	`case CMD4GReadDeviceTime:`
	`  [weakself BPMBLEManagerResponseReadDeviceTime:model];`
	`  break;`

5.9. [4G Dedicated] Write device Time to BPM:

	- (void)syncTiming;
Definition	Write device Time to BPM: After the device is connected successfully, the time will be automatically synchronized.
CMD	CMD4GSyncTiming
model	MicroLifeDataModel

	`case CMD4GSyncTiming: {`
	`  MicroLifeDataModel *data = model;`
	`  [weakself`
	`  BPMBLEManagerResponseWriteDeviceTime:data.success`
	`  .boolValue];`
	`}`
	`  break;`

5.10. 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	MicroLifeBloodPressureDRecord (with measurement data) MicroLifeDataModel (without measurement data)

	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];
	}
	}

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

	- (void)readLastData;(3G) - (BOOL)readLastData;(4G)
Definition	Clear last 1 data of the BPM: [Attention] will not respond!
CMD	CMD3GClearLastData CMD4GClearLastData
model	MicroLifeDataModel

	case CMD3GClearLastData: {
	MicroLifeDataModel *data = model;
	[weakself
	BPMBLEManagerResponseClearLastData:data.success
	.boolValue];
	}
	break;

	case CMD4GClearLastData: {
	MicroLifeDataModel *data = model;
	[weakself
	BPMBLEManagerResponseClearLastData:data.success
	.boolValue];
	}
	break;

5.12. Read device SN from BPM:

	- (BOOL)readSerialNumber;
Definition	Read device SN from BPM
CMD	CMD4GReadSerialNumber
model	MicroLifeDeviceInfo

	case CMD4GReadSerialNumber: {
	MicroLifeDeviceInfo *deviceInfo = model;
	NSLog(@"Serial Number:%@",deviceInfo.sn);
	}
	break;

6. Usage:

6.1. independent management mode (Multiple devices are connected at the same time):

6.1.1. Single-instance Parallax device:

```
self.bp4g = [MicroLife4GBP shareWhithAuthorizationkey:SDKkey_BPM];
```

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

```
[self.bp4g 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.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;
    }
}];
```

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

```
[self.bp4g 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.bp4g = [MicroLife4GBP shareWhithAuthorizationkey:SDKkey_BPM];
```

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. When the mobile phone's 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.sdk device:@[self.bp3g,self.bp4g]];
```

6.2.7. Monitoring command response status: Send commands according to usage needs, and obtain corresponding information through CMD.

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

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

```
[self.bp4g 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 to 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 to cancel the automatic scanning mechanism.

```
[self.bp3g 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.bp3g setAutoConnect:NO];
```

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

```
[self.sdk getScanDeviceBlock:^(id  _Nonnull device) {
    NSString *log = [NSString stringWithFormat:@"ScanDevice
        Name：%@ UUID：%@ mac：%@",[device name],[device
        UUID],[device mac]];
    [weakself addLogWhitText:log];
    if ([device isEqual:self.bp3g]) {
        [self.bp3g connectDevice];
    }
} CancelScanBlock:^() {
    [weakself addLogWhitText:@"Cancel Scan"];
}];
```

7. V1.x Migration to V2.x:

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 manager of the original V1.x version.

```
self.aBPMBLEManager = [BPMBLEManager
    shareInstanceWhithAuthorizationkey:SDKkey_BPM];
self.aBPMBLEManager.dataResponseDelegate = self;
```

7.3. - (void)BPMBLEManagerCellPhoneBluetoothDidUpdateState:(MicroLifeBLEState)state, change MicroLifeBLEState to CBManagerState, and inherit it in getDidUpdateStateBlock:^(CBManagerState state).

7.4. - (void)BPMBLEManagerDidDiscoverBluetoothDeviceMacAddress:(nonnull NSData *)macAddress Name:(nonnull NSString *)name RSSI:(nonnull NSNumber *)RSSI , replaced by ScanDeviceBlock:^(id _Nonnull device).

7.5. - (void)BPMBLEManagerDidConnectDevice,
- (void)BPMBLEManagerDidDisconnectDevice,
- (void)BPMBLEManagerDidFailToConnectDevice have been replaced by ConnectDeviceStateBlock:^(id _Nonnull device, CBCentralManager * _Nonnull central, CBPeripheral * _Nonnull peripheral, CBPeripheralState state, NSError * _Nonnull error).

```
[self.bp4g 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. - (void)BPMBLEManagerRespondTimeOut has been replaced by getValidationErrorBlock:^(NSString * _Nonnull item, NSString * _Nonnull info, NSData * _Nonnull value).

```
[self.bp4g getValidationErrorBlock:^(NSString * _Nonnull item,
    NSString * _Nonnull info, NSData * _Nonnull value) {

}];
```

7.7. BPMDataResponseDelegate has been getReadDataModelBlock:^(NSInteger CMD, id _Nonnull model, NSError * _Nonnull error), and the corresponding response data is obtained from CMD.

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

7.8. - (void)BPMBLEManagerResponseReadAllHistorys:(nonnull MicroLifeDRecord *)data, MicroLifeDRecord is replaced by MicroLifeBloodPressureDRecord, MicroLifeCurrentAndMData is replaced by MicroLifeBloodPressureCurrentAndMData, CMD is equal to CMD4GReadHistorys.

```
case CMD4GReadHistorys:
    [weakself BPMBLEManagerResponseReadHistory:model];
    break;
```

7.9. - (void)BPMBLEManagerResponseReadUserAndVersionData:(nonnull MicroLifeUserInfo *)user VersionData:(nonnull MicroLifeDeviceInfo *)verData, CMD is equal to CMD4GReadUserAndVersionData.

```
case CMD4GReadUserAndVersionData: {
    MicroLifeUserInfo *userInfo = [model
        valueForKey:@"UserInfo"];
    MicroLifeDeviceInfo *version = [model
        valueForKey:@"Version"];
    [weakself
        BPMBLEManagerResponseReadUserAndVersionData:userIn
        fo VersionData:version];
}
    break;
```

7.10. - (void)BPMBLEManagerResponseClearHistory:(BOOL)isSuccess, CMD is equal to CMD4GClearAllHistorys.

```
case CMD4GClearAllHistorys: {
    MicroLifeDataModel *data = model;
    [weakself
        BPMBLEManagerResponseClearHistory:data.success
        .boolValue];
}
    break;
```

7.11. - (void)BPMBLEManagerResponseReadDeviceTime:(nonnull MicroLifeDeviceInfo *)deviceInfo, CMD is equal to CMD4GReadDeviceTime.

```
case CMD4GReadDeviceTime:
    [weakself BPMBLEManagerResponseReadDeviceTime:model];
    break;
```

7.12. - (void)BPMBLEManagerResponseWriteDeviceTime:(BOOL)isSuccess, CMD is equal to CMD4GSyncTiming.

```
case CMD4GSyncTiming: {
    MicroLifeDataModel *data = model;
    [weakself
        BPMBLEManagerResponseWriteDeviceTime:data.success
        .boolValue];
}
    break;
```

7.13. - (void)BPMBLEManagerResponseWriteUserID:(BOOL)isSuccess, CMD is equal to CMD4GWriteUser.

```
case CMD4GWriteUser: {
    MicroLifeDataModel *data = model;
    [weakself
        BPMBLEManagerResponseWriteUserID:data.success
        .boolValue];
}
    break;
```

7.14. - (void)BPMBLEManagerResponseReadDeviceInfo:(nonnull MicroLifeDeviceInfo *)deviceInfo, CMD is equal to CMD4GReadDeviceInfo.

```
case CMD4GReadDeviceInfo:
    [weakself BPMBLEManagerResponseReadDeviceInfo:model];
```

7.15. Read Serial Number, CMD is equal to CMD4GReadSerialNumber.

```
case CMD4GReadSerialNumber: {
    MicroLifeDeviceInfo *deviceInfo = model;
    NSLog(@"Serial Number:%@",deviceInfo.sn);
}
    break;
```

8. Demo App:
    8.1. Navigation page:
        8.1.1. A: Body Temperature SDK, function display of forehead temperature SDK.
        8.1.2. B: Blood Pressure SDK, blood pressure machine SDK function display.
        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 logs to MicroLife.
        8.1.8. H: One For All, automatic multi-device connection display, independent management mode (multi-device connection 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. Blood Pressure SDK interface description:

8.2.1. A: Log display area.

8.2.2. B: Function command area, which can be slid up and down.

8.2.3. For related code, refer to BPMViewController.

8.2.4. Instructions for use:

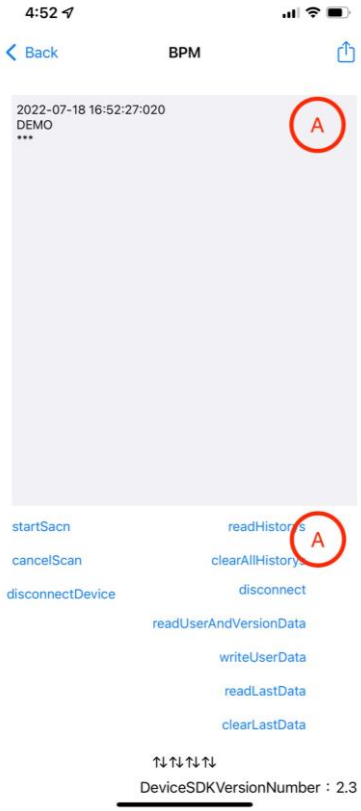
8.2.4.1. Entering the display page will automatically perform a device search.

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, the SDK will automatically send "Write device Time to BPM" and "Read user ID and version data from BPM".

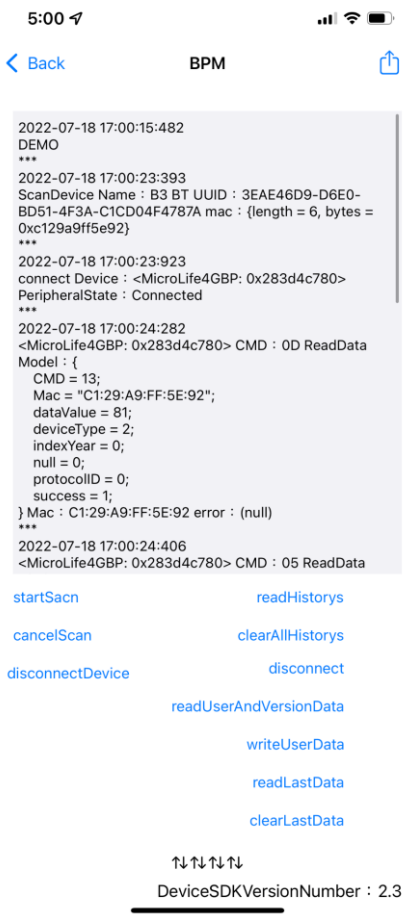
8.2.4.4. After the device is connected, select the function operation in the device function command area.

8.2.4.5. After the device is connected, if no command is sent for more than 120 seconds, it will automatically disconnect.



9. Operation

9.1. Search & Connect:

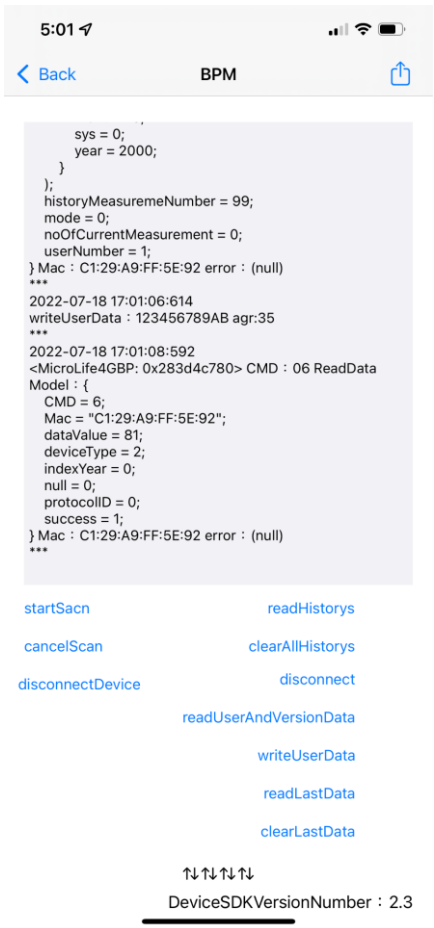


1. After entering the display page, the SDK will automatically start searching and connecting to the matching device.
2. ScanDevice Name: B3 BT UUID: 3EAE46D9-D6E0-BD51-4F3A-C1CD04F4787A mac: {length = 6, bytes = 0xc129a9ff5e92}
3. connect Device: <MicroLife4GBP: 0x283d4c780> PeripheralState: Connected

9.2. Pairing:

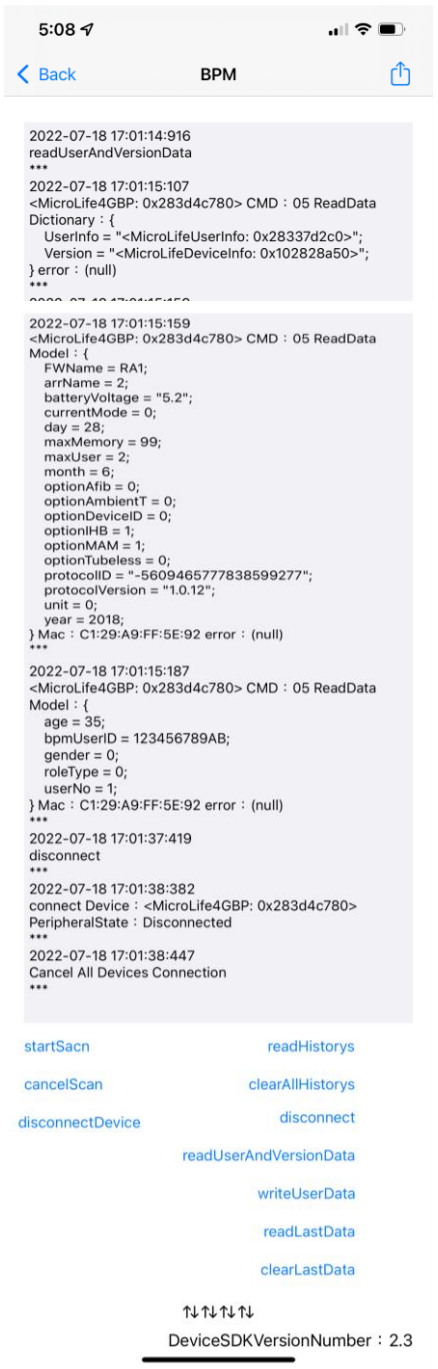
1. If the device needs Bluetooth pairing, it will pop up the Bluetooth pairing dialog window.
2. After the pairing is completed, select the test command.

9.3. Test command (eg: Write a new user ID):



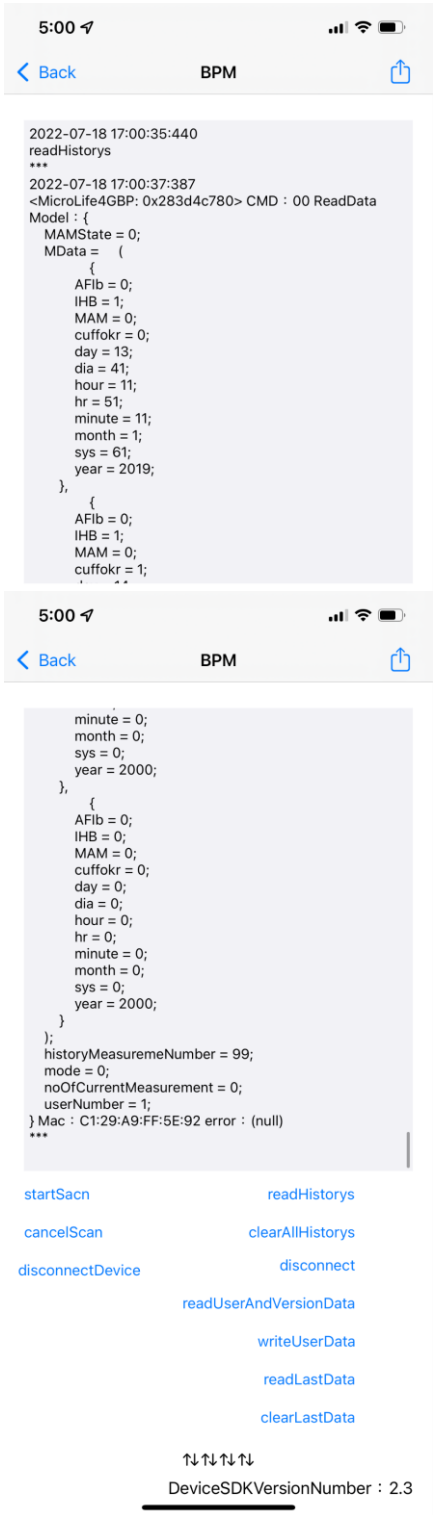
1. Select the test command: Write a new user ID.
2. writeUserData: 123456789AB agr:35. User ID is Ascii code.
3. <MicroLife4GBP: 0x283d4c780> CMD: 06 ReadData Model: {
CMD = 6;
Mac = "C1:29:A9:FF:5E:92";
= 81;
deviceType = 2;
indexYear = 0;
null = 0;
protocolID = 0;
success = 1;
} Mac: C1:29:A9:FF:5E:92 error: (null)
: Write status response.
4. The newly written user ID can be checked with "Read user ID and version data".

9.4. Test command (eg: Read user ID and version data):



1. Select the test command: Read user ID and version data.
2. Response: <MicroLife4GBP: 0x283d4c780> CMD: 05 ReadData Dictionary: {
UserInfo = "<MicroLifeUserInfo: 0x28337d2c0>";
Version = "<MicroLifeDeviceInfo: 0x102828a50>";
} error: (null)
3. MicroLifeUserInfo: User Info
4 . MicroLifeDeviceInfo: Device information

9.5. test command (eg: Read all history data):



1. Select the test command: Read all history data.
2. <MicroLife4GBP: 0x283d4c780> CMD: 00 ReadData Model: {
MAMState = 0;
MData = (
{
AFIb = 0;
IHB = 1;
MAM = 0;
cuffokr = 0;
day = 13;
dia = 41;
hour = 11;
hr = 51;
minute = 11;
month = 1;
sys = 61;
year = 2019;
},
{...},
...,
{...}
);
currentData = (
{…},
{…},
{…}
);
historyMeasuremeNumber = 99;
mode = 0;
noOfCurrentMeasurement = 0;
userNumber = 1;
} Mac: C1:29:A9:FF:5E:92 error: (null)

10. Update record:
    - 10.1. Difference V2.x vs V1.x:
        - 10.1.1. All request responses are changed from delegate mode to block mode.
        - 10.1.2. Separate general-purpose tools and Microlife 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