

Demo APP for MicroLifeDeviceSDK (Android)

Table of Contents

Chapter 1	Development Environment
Chapter 2	Entry Point and Bluetooth LE Protocol
Chapter 3	Bluetooth LE Protocol APIs
Chapter 4	PFM APIs
Chapter 5	User Interface of Demo App
Chapter 6	Functionality of Demo App
	Appendix

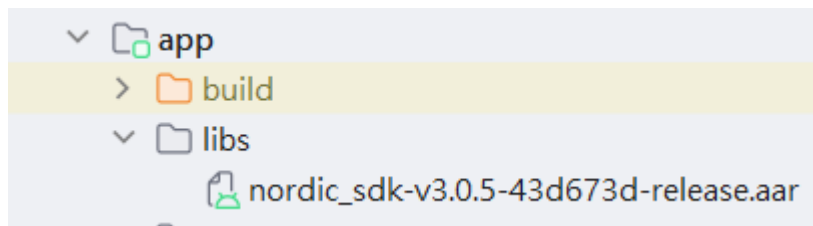
Chapter1 Development Environment

1.1 The supported SDK version is as follow:

```
android {
    namespace "com.mlc.nordic_sdk"
    compileSdkVersion 36

    defaultConfig { DefaultConfig it ->
        applicationId "com.mlc.nordic_sdk"
        minSdk 29
        targetSdk 36
        versionCode 1
        versionName "2.2.0"
    }
}
```

1.2 Add the library “nordic_sdk_vxxx-xxxxxxx.aar” into the “libs” directory.



1.3 In the “build.gradle”, add the description as bellows.

```
//aar (sdk required)
implementation fileTree(dir: "libs", include: ["*.aar", "*.jar"], exclude: [])

//nordic scan (sdk required)
implementation libs.scanner

//nordic connect (sdk required)
implementation libs.ble
implementation libs.ble.ktx
implementation libs.ble.common
implementation libs.ble.livedata

//open csv (sdk required)
implementation (libs.opencsv)

//livedata
implementation libs.runtime.livedata
```

Chapter2 Entry Point and Bluetooth LE Protocol

The “ChoseActivity” is the entry point of the sample application. The “PFMAActivity” is dedicated to the device of general Peak Flow Device (Bluetooth LE).

```
        android:theme="@style/Theme.Nordic_SDK" />
    <activity
        android:name=".activity.ChoseActivity"
        android:exported="true"
        android:theme="@style/Theme.Nordic_SDK">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity
        android:name=".activity.BPMActivity"
        android:exported="true"
        android:theme="@style/Theme.Nordic_SDK" />
    <activity
        android:name=".activity.SP02Activity"
        android:exported="true"
        android:theme="@style/Theme.Nordic_SDK" />
    <activity
        android:name=".activity.PFMAActivity"
        android:exported="true"
        android:theme="@style/Theme.Nordic_SDK" />
    <activity
        android:name=".activity.ThermoActivity"
        android:exported="true"
        android:theme="@style/Theme.Nordic_SDK" />
    <activity
        android:name=".activity.BaseActivity"
        android:exported="true"
        android:theme="@style/Theme.Nordic_SDK"/>
    <activity
        android:name=".activity.BGMAActivity"
        android:exported="true"
        android:theme="@style/Theme.Nordic_SDK"/>
    <activity
        android:name=".activity.WBPAActivity"
        android:exported="true"
        android:theme="@style/Theme.Nordic_SDK"/>
</application>
```

2.1 Initialize the instance “bluetoothManager”. This is to fulfill Bluetooth LE features and connection sequence.

```
bluetoothManager = BluetoothManager.getInstance( activity = this, listener = this)
```

2.1.1 OnIMBluetoothLEListener: For Bluetooth progress.

```
interface OnIMBluetoothLEListener {
    3 Usages  5+ Implementations
    fun onScanResult(
        device: BluetoothDevice,
        deviceName: String,
        deviceType: DeviceType?,
        macAddress: String?
    )

    5+ Implementations
    fun onConnectionState(connectState: ConnectState)

    5+ Implementations
    fun onConnectionState(connectState: ConnectState, state: Int)

    3 Usages  5+ Implementations
    fun onReceivedBleDataResult(data: List<Byte>, head: Int? = null)

    2 Usages  5+ Implementations
    fun onResponseSWRevision(swRevision: String)

    2 Usages  5+ Implementations
    fun onResponseFWRevision(fwRevision: String)

    2 Usages  5+ Implementations
    fun onResponseHWRevision(hwRevision: String)

    2 Usages  5+ Implementations
    fun onBtStateChanged(isEnable: Boolean)

    2 Usages  5+ Implementations
    fun onRetryFailed(msg: String)
}
```

2.2 Initialize the instance “pfmProtocol”. This is to fulfill retrieving PFM commands and parsing data from the PFM device.

```
// Input protocol key
17 Usages
private var pfmProtocol: PFMProtocol? = PFMProtocol.getInstance(sdkid = "")
```

2.3 PFMProtocol. OnDataResponseListener: For PFM Protocol progress

```
interface OnDataResponseListener {
    2 Usages 2 Implementations
    fun onResponsePFMReadHistory(dRecord: DRecordPFM?)
    2 Usages 2 Implementations
    fun onResponsePFMClearAllHistory(isSuccess: Boolean)
    1 Usage 2 Implementations
    fun onResponseDisconnect(isSuccess: Boolean)
    2 Usages 2 Implementations
    fun onResponsePFMReadUserIdAndVersionData(userIdAndVersionData: UserIdAndVersionData)
    2 Usages 2 Implementations
    fun onResponsePFMWriteNewUser(isSuccess: Boolean)
    3 Usages 2 Implementations
    fun onResponsePFMReadLastData(dRecord: DRecordPFM?)
    2 Usages 2 Implementations
    fun onResponsePFMClearLastData(isSuccess: Boolean)
    3 Usages 2 Implementations
    fun onResponsePFMReadDeviceTime(deviceTime: DeviceTime?)
    2 Usages 2 Implementations
    fun onResponsePFMWriteDeviceTime(isSuccess: Boolean)
    2 Usages 2 Implementations
    fun onResponsePFMReadSerialNumber(serialNumber: String?)
    2 Usages 2 Implementations
    fun onResponsePFMWriteBestValue(isSuccess: Boolean)
    1 Usage 2 Implementations
    fun onResponsePFMCheckMode(mode: Int, data: Int)
    2 Usages 2 Implementations
    fun onResponsePFMStartMeasurement(isDeviceReady: Boolean)
    2 Usages 2 Implementations
    fun onResponsePFMReadWaveform(pfmWaveForm: PFMWaveForm)
}
```

Chapter3 Bluetooth LE Protocol APIs

3.1 Instance of Bluetooth LE Protocol

3.1.1 Interface :

	<pre>fun getInstance(activity: Activity, listener: OnIMBluetoothLEListener? = null): BluetoothManager?</pre>
Definition	Initialize Bluetooth LE Protocol
Parameter	activity : name of activity or this listener : get the Bluetooth status
	<pre>bluetoothManager = BluetoothManager.getInstance(activity = this, listener = this)</pre>

3.2 Read and Write command to Bluetooth Device

3.2.1 Interface :

	suspend fun writeCommand(data: ByteArray?)
Definition	write command to Bluetooth Device
Parameter	data : ByteArray command

3.2.2 Delegate :

	<pre>override fun onReceivedBleDataResult(data: List<Byte>, head: Int?)</pre>
Definition	This is to get List<Byte> from Bluetooth device.

3.3 Connection State and Result

3.3.1 Interface :

	<pre> override fun onScanResult(device: BluetoothDevice, deviceName: String, deviceType: DeviceType?, macAddress: String?) </pre>
Definition	This is to get Bluetooth information of devices which discovered in the vicinity.
Parameter	<pre> enum class DeviceType { 4 Usages ALL, MLC_ALL, 6 Usages MLC_BPM3G, MLC_BPM4G, MLC_BPM5G, 4 Usages MLC_Thermo, MLC_SP02, MLC_PF2, 5 Usages MLC_WS, MLC_BGM, MLC_WBP, 1 Usage WAG_ALL, WAG_BPM4G } </pre>

	override fun onConnectionState(connectState: ConnectState)
Definition	The “onConnectionState()” is to monitor the status of connection.
Parameter	<pre>enum class ConnectState { 1 Usage StartScan, StopScan, ScanFinished, Connected, DeviceReady, 4 Usages Disconnect, ConnectFailed, 2 Usages Bonded, BondNone, Bonding, ERROR_133 }</pre>

	override fun onReceivedBleDataResult(data: List<Byte>, head: Int?)
Definition	This is to get List<Byte> from Bluetooth device.

	override fun onResponseSWRevision(swRevision: String)
Definition	This is used to obtain the software revision of the Bluetooth device.

	override fun onResponseFWRevision(fwRevision: String)
Definition	This is used to obtain the hardware revision of the Bluetooth device.

	override fun onBtStateChanged(isEnable: Boolean)
Definition	This is to monitor the state of Enabled or Disabled.

	override fun onRetryFailed(msg: String)
Definition	Return if sending the command fails.

3.4 Device scanning or discovery

3.4.1 Interface :

	fun startScan(scanDeviceType: DeviceType)	
Definition	The “startScan()” is for device scanning or discovery. The result will be shown with the “onScanResult”.	
Parameter	<pre>enum class DeviceType { 4 Usages ALL, MLC_ALL, 6 Usages MLC_BPM3G, MLC_BPM4G, MLC_BPM5G, 4 Usages MLC_Thermo, MLC_SP02, MLC_PF2, 5 Usages MLC_WS, MLC_BGM, MLC_WBP, 1 Usage WAG_ALL, WAG_BPM4G }</pre>	

	fun stopScan()	
Definition	Terminate the scanning process.	

3.5 Disconnection

	fun disconnectGatt()	
Definition	Disconnect device.	

Chapter4 PFM APIs

4.1 Instance of PFM Protocol

4.1.1 Interface :

	<pre>fun getInstance(sdkid: String, listener: OnDataResponseListener?): PFMPProtocol?</pre>
Definition	Initialize PFM Protocol
Parameter	<p>sdkid : SDK ID of designated device</p> <p>listener : This is to obtain the parsed PFM information.</p>
	<pre>// Input protocol key private var pfmProtocol: PFMPProtocol? = PFMPProtocol.getInstance(sdkid = "", listener = this)</pre>

4.2 Read all history data from PFM :

4.2.1 Interface :

	<pre>fun readAllHistoryData(selectUser: Long): ByteArray</pre>
Definition	Read all history data.

4.2.2 Delegate :

	<pre>fun onResponsePFMReadHistory(dRecord: DRecordPFM?)</pre>
Parameter	<pre>public final data class DRecordPFM(public final var mode: Int? = null, public final var historyMeasurementTimes: Int? = null, public final var userNumber: Int? = null, public final var mamVersion: Int? = null, public final var mData: List<MData?>? = null)</pre>

4.3 Clear all history data of PFM :

4.3.1 Interface :

	fun clearAllHistoryData(selectUser: Long): ByteArray
Definition	Clear all history data.

4.3.2 Delegate :

	fun onResponsePFMClearAllHistory(isSuccess: Boolean)
Parameter	success : True of False

4.4 Read user id and version data from PFM :

4.4.1 Interface :

	fun readUserIdAndVersionData(): ByteArray
Definition	Read user id and version data.

4.4.2 Delegate :

	fun onResponsePFMReadUserIdAndVersionData(userIdAndVersionData: UserIdAndVersionData)	
Parameter	<pre> public final data class UserIdAndVersionData(public final var currentUserNo: Int? = null, public final var user1_id: String? = null, public final var user1_age: Int? = null, public final var user2_id: String? = null, public final var user2_age: Int? = null, public final var fwVersion: String? = null, public final var vYear: Int? = null, public final var vMonth: Int? = null, public final var vDay: Int? = null, public final var maxUser: Int? = null, public final var maxMemory: Int? = null, public final var deviceBatt: Float? = null, public final var p_id: String? = null, public final var arrName: String? = null, public final var currentMode: Int? = null) </pre>	

4.5 Write new user to PFM :

4.5.1 Interface :

	<pre>fun writeNewUser(selectUser: Long, id: String, age: Long): ByteArray</pre>
Definition	Write new user.

4.5.2 Delegate :

	<pre>fun onResponsePFMWriteNewUser(isSuccess: Boolean)</pre>
Parameter	success : True of False

4.6 Read last data from PFM :

4.6.1 Interface :

	<pre>fun readLastData(): ByteArray</pre>
Definition	Read last data.

4.6.2 Delegate :

	<pre>fun onResponsePFMReadLastData(dRecord: DRecordPFM?)</pre>
Parameter	<pre>public final data class DRecordPFM(public final var mode: Int? = null, public final var historyMeasurementTimes: Int? = null, public final var userNumber: Int? = null, public final var mamVersion: Int? = null, public final var mData: List<MData?>? = null)</pre>

4.7 Clear last data from PFM :

4.7.1 Interface :

	fun clearLastData(): ByteArray
Definition	Clear last data.

4.7.2 Delegate :

	fun onResponsePFMClearLastData(isSuccess: Boolean)
Parameter	success : True of False

4.8 Read device time from PFM :

4.8.1 Interface :

	fun readDeviceTime(): ByteArray
Definition	Read device time.

4.8.2 Delegate :

	fun onResponsePFMReadDeviceTime(deviceTime: DeviceTime?)
Parameter	<pre> public final data class DeviceTime(public final var isTimeReady: Boolean? = null, public final var year: Int? = null, public final var month: Int? = null, public final var day: Int? = null, public final var hour: Int? = null, public final var minute: Int? = null, public final var second: Int? = null) </pre>

4.9 Write device time to PFM :

4.9.1 Interface :

	fun writeDeviceTime(): ByteArray
Definition	Write device time.

4.9.2 Delegate :

	fun onResponsePFMWriteDeviceTime(isSuccess: Boolean)
Parameter	success : True of False

4.10 Read serial number from PFM :

4.10.1 Interface :

	fun readSerialNumber(): ByteArray
Definition	Read serial number.

4.10.2 Delegate :

	fun onResponsePFMReadSerialNumber(serialNumber: String?)
Parameter	success : True of False

4.11 Read best value from PFM :

4.11.1 Interface :

	fun readBestValue (): ByteArray
Definition	Read best value.

4.11.2 Delegate :

	fun onResponsePFMReadBestValue(bestValue: Int, highValue: Int)
Parameter	bestValue: Int, highValue: Int

4.12 Write best value to PFM :

4.12.1 Interface :

	fun writeBestValue(bestValue: Int): ByteArray
Definition	Write best value.

4.12.2 Delegate :

	fun onResponsePFMWriteBestValue(isSuccess: Boolean)
Parameter	success : True of False

4.13 Check mode from PFM :

4.13.1 Interface :

	fun checkMode(): ByteArray
Definition	check mode.

4.13.2 Delegate :

	fun onResponsePFMCheckMode(mode: Int, data: Int)
Parameter	Mode = 0 : Normal mode Mode = 1 : Waveform mode Data = 0 : Have not new data Data = 1 : Have new data

4.14 Start measurement from PFM :

4.14.1 Interface :

	fun startMeasurement(): ByteArray
Definition	Start measurement.

4.14.2 Delegate :

	fun onResponsePFMStartMeasurement(isDeviceReady: Boolean)
Parameter	success : True or False

4.15 Read wave form from PFM :

4.15.1 Interface :

	fun readWaveform(): ByteArray
Definition	Read wave form.

4.15.2 Delegate :

	fun onResponsePFMReadWaveform(pfmWaveForm: PFMWaveForm)
Parameter	<pre> public final data class PFMWaveForm(public final var mData: List<Int?>? = null, public final var mResult: MResult? = null) public final data class MResult(public final var pef: Int? = null, public final var fev1: Int? = null, public final var fvc: Int? = null, public final var year: Int? = null, public final var month: Int? = null, public final var day: Int? = null, public final var hour: Int? = null, public final var minute: Int? = null, public final var second: Int? = null) </pre>

4.16 Solve Data Result

4.16.1 Interface :

	fun solveDataResult(data: String)
Definition	Parses the List<Byte> received from the Bluetooth device.

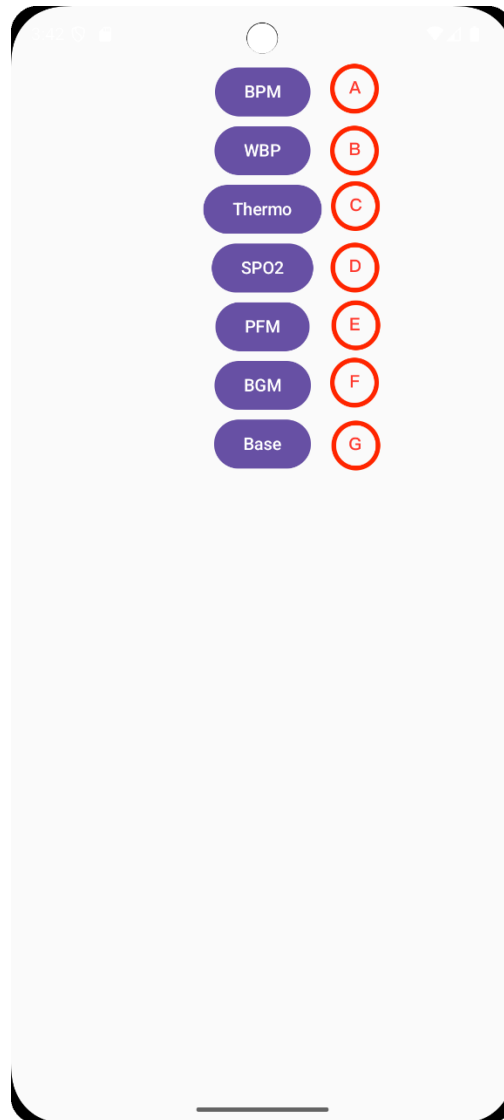
4.16.2

	fun solveDataResult(data: List<Byte>)
Definition	Parses the List<Byte> received from the Bluetooth device.

Chapter5 User Interface of Demo App

5.1 Getting Started :

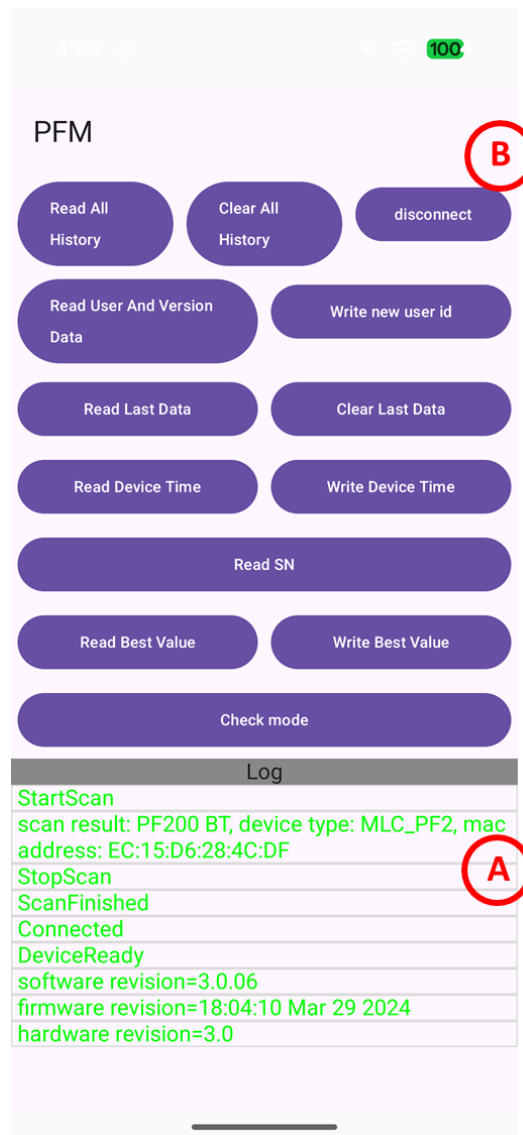
Start the app and then select the button “PFM” / “E” to communicate with the designate device PFM.



5.2 Operation Sequence :

The scanning (discovery) is automatically run to discover devices in the vicinity. If a device is bonded, it will be connected accordingly. If not, the “bindingDevice” can be used to run bonding process.

5.3 GUI Layout :




5.3.1 Region A : The log window is used to display information about communication handshake between App and device.

5.3.2 Region B : This part is to communicate with the device PFM by different functions / commands such as data transferring, synchronization and so on.

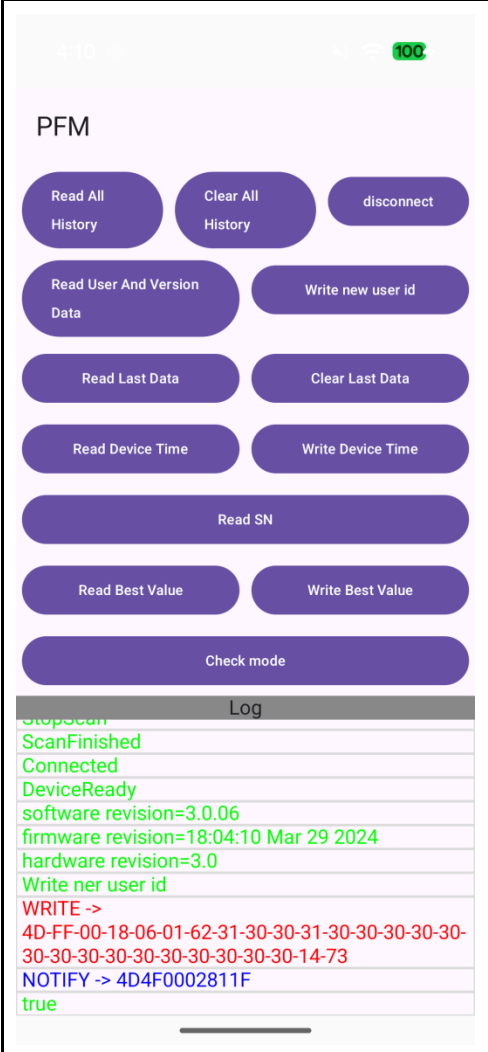
5.3.3 Refer to “PFMActivity” from the demo application (sample code) to get more detailed.

Chapter6 Functionality of Demo App

6.1 Bluetooth authorization :

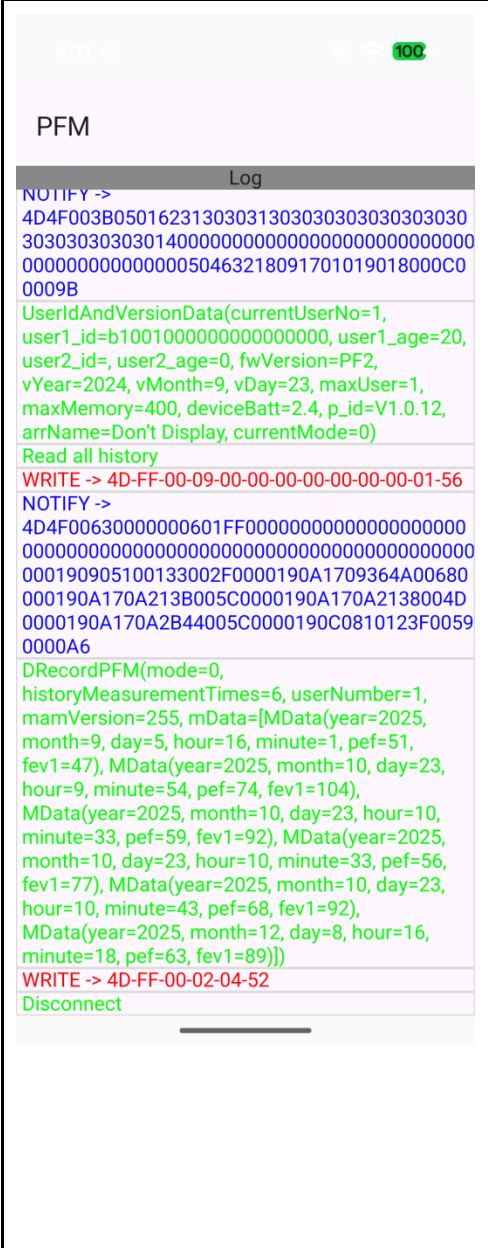
 A screenshot of an Android smartphone screen showing a permission dialog. At the top, the status bar shows the time 9:17, signal strength, Wi-Fi, and battery at 95%. A dark purple banner at the top of the dialog says "Request Permission". The dialog itself is a light purple rounded rectangle with a blue diamond icon at the top. The text inside asks: "Allow Microlife SDK to find, connect to, and determine the relative position of nearby devices?". At the bottom are two buttons: "Allow" and "Don't allow".	<p>1. Request for Bluetooth permission.</p>
--	--

6.2 Command: Write a new user ID to PFM:

	<ol style="list-style-type: none">1. The command “WRITE USER” is to write a new user ID to PFM.2. The log “WRITE : write user“ and “Write : userID:b1001” are indicated that the App sends a command with an user ID. The ID is made up of ASCII code.3. The log “PFM:WriteUser -> isSuccess = true” means that the writing/ sending procedure is successful.4. The red part is the command and communication protocol that is sent to device. The blue part is notification with the raw data from PFM via Bluetooth.
--	--

- ### 6.3 Command: Read user ID and version data from PFM:

6.4 Command: Read all history from PFM

 <p>The screenshot shows the PFM application interface. At the top, there's a status bar with a battery icon at 100%. Below it, the title 'PFM' is displayed. A 'Log' section is visible, containing several entries: a 'NOTIFY ->' command with a long hexadecimal string; a 'UserAndVersionData' log entry with various parameters like userNo, user1_id, user1_age, user2_id, user2_age, fwVersion, vYear, vMonth, vDay, maxUser, maxMemory, deviceBatt, p_id, arrName, and currentMode; a 'Read all history' command; a 'WRITE ->' command with a hexadecimal string; another 'NOTIFY ->' command with a long hexadecimal string; a 'DRecordPFM' log entry with parameters like mode, historyMeasurementTimes, userNumber, mamVersion, and an array of MData objects; a 'WRITE ->' command with a hexadecimal string; and finally a 'Disconnect' command.</p>	<ol style="list-style-type: none"> 1. The button “Read all history” is to get all history data. 2. The red part is the command and communication protocol that is sent to device. The blue part is notification with the raw data from PFM via Bluetooth. The green part is the result after decoding with the raw data. 3. The log “DRecordPFM(mode=0, historyMeasurementTimes=6, userNumber=1, mamVersion=255, mData=[MData(year=2025, month=9, day=5, hour=16, minute=1, pef=51, fev1=47), MData(year=2025, month=10, day=23, hour=9, minute=54, pef=74, fev1=104),...]” is included PFM readings. 4. Each PFM reading has its own measurement date & time, pef and fev1. For instance, the above-mentioned reading is DateTime 2025/9/5 16:01, PEF 51, FEV1 47.
---	---

Appendix

CurrentAndMData Structure / Parameter				
Property	Device	Type	Value	Description
AA=0	WBP	Integer	0, 1	Anti-artifact Detected
ABPM=0	WBP	Integer	0, 1	Ambulatory Blood Pressure Monitoring
AFIb=false	3G/4G	Boolean	false , true	Atrial Fibrillation Detection
AM=false	WBP	Boolean	false , true	Data measured at night with diagnostic mode (e.g., 4:00~12:00)
CPP=0	WBP	Integer	0 ~ 255	Central Pulse Pressure
CSBP=0	WBP	Integer	0 ~ 255	Carotid Systolic Blood Pressure
IHB=false	3G/4G	Boolean	false , true	Irregular Heartbeat Detection
LB=0	WBP	Integer	0, 1	Low Battery
MAM=0	3G/4G	Integer	0 ~ 3	Microlife Aerge Mode
MAP=0	WBP	Integer	0 ~ 255	Mean Arterial Pressure
MCBP=0	WBP	Integer	0 ~ 65535	Mean Central Blood Pressure
PM=false	WBP	Boolean	false , true	Data measured in the morning with diagnostic mode

				(e.g., 18:00~24:00)
PVR=0	WBP	Integer	0 ~ 65535	Pulse Variation Rate
SM=0,	WBP	Integer	0, 1	Start of a Manual Measurement
arr=false	3G/4G	Boolean	false , true	Detection of PAD or Afib
condition=0	WBP	Integer	0 ~ 24	Condition
Cuffokr=0	3G/4G	Integer	0, 1	Whether the wristband is tight: 3G detection mode: 2 - No detection, 4G detection mode: 0 - No tightness, 1 - There is tightness
day=7	WBP, 3G/4G	Integer	0 ~ 31	Date
deviceMode=58	WBP, 3G/4G	Integer	0x31 3G BPM, 0x3A 4G BPM, 0X51 WBP HomeA	Device type
dia=69	WBP, 3G/4G	Integer	0 ~ 255	Diastolic
diagnostic=false	WBP	Boolean	false , true	Data measured in diagnostic mode
errorCode=0	WBP	Integer	0 ~ 255	Error Code
hour=10	WBP, 3G/4G	Integer	0 ~ 24	Hour

hr=82	WBP, 3G/4G	Integer	0 ~ 255	Pulse
indexYear=0	WBP	Integer	0 ~ 99	offset Year
isFor3G=false	3G/4G	Boolean	false , true	is for 3G
minute=12	WBP, 3G/4G	Integer	0 ~ 59	Minute
month=2,	WBP, 3G/4G	Integer	1 ~ 12	Month
resultCode=0	WBP	Integer	0x01, 0x02, 0x03, 0x05, 0x42, 0x50, 0x51, 0x52	Result Code
systole=114	WBP, 3G/4G	Integer	0 ~ 255	Systolic
usual=false	WBP	Boolean	false , true	Data measured in usual mode
year=2024	WBP, 3G/4G	Integer	WBP: 2000~ 2050, 3G/ 4G: 2000 ~ 2048	Year