

IOS PO SDK Documentation

1. Relevant files and frameworks

- (1) Import the following PO SDK files: POHeader.h、POMacroFile.h、PO3.h、PO3Controller.h、User.h、iHealthLibrary(1.0.7).a, supports iOS 6.0 and above.

(2) Frameworks

▼ Custom iOS Target Properties










Key	Type	Value
Bundle versions string, short	String	1.0
Bundle identifier	String	com.zhang.xxx.\${PRODUCT_NAME:rfc1034identifie
InfoDictionary version	String	6.0
Main storyboard file base name	String	Main
Bundle version	String	1.0
▼ Required background modes	Array	(2 items)
Item 0	String	App communicates using CoreBluetooth
Item 1	String	App communicates with an accessory
▼ Supported external accessory protocols	Array	(6 items)
Item 0	String	com.jiuan.P930
Item 1	String	com.jiuan.BPV21
Item 2	String	com.jiuan.BPV20
Item 3	String	com.ihealth.sc221
Item 4	String	com.jiuan.BGV30
Item 5	String	com.jiuan.BGV31

(3) Configuration

Add an “item” in “Info”

Add two new items in “Required background modes” , App communicates with an accessory 、App communicates using CoreBluetooth

▼ Link Binary With Libraries (9 items)

Name	Status
 MediaPlayer.framework	Required ⇅
 SystemConfiguration.framework	Required ⇅
 Accelerate.framework	Required ⇅
 ExternalAccessory.framework	Required ⇅
 CoreBluetooth.framework	Required ⇅
 CoreGraphics.framework	Required ⇅
 iHealthLibrary(1.0.7).a	Required ⇅
 UIKit.framework	Required ⇅
 Foundation.framework	Required ⇅
+ — Drag to reorder frameworks	

2. Operation Procedure

(1) PO3 instructions

Register device PO3: `PO3ConnectNoti;`

Initialize PO3 controller class:

```
PO3Controller *po3Controller = [PO3Controller sharedInstance];
```

Access control class instance after receiving PO3ConnectNoti
NSArray *po3Array = [po3Controller getAllCurrentPO3Instance];
PO3 *po3Instance = [po3Array objectAtIndex:0];
Use po3Instance to call PO3 related communication methods.

3. PO3 Interface Instructions

(1) Sync time

Only after verification through this interface can we move onto using other API's.

-(void)commandCreatePO3User:(User *)tempUser

Authentication:(BlockUserAuthentication)disposeAuthenticationBlock

DisposeResultBlock:(DisposeSynchronousTimeFinishBlock)disposeSynchronousTimeFinishBlock
DisposeErrorBlock:(DisposePO3ErrorBlock)disposeErrorBlock;

Input Parameters:

tempUser, includes properties: clientID, clientSecret, userID。

userID, either email or mobile phone number (mobile phone number not yet supported).

ClientID and clientSecret, the only identification for users of the SDK, requires registration from iHealth administrator, please email: lvjincan@jiuan.com for more information

Return Parameters:

disposeAuthenticationBlock: The return parameters of 'userid', 'clientID', and 'clientSecret' after verification。

The interpretation for the verification:

UserAuthen_RegisterSuccess: New-user registration succeeded.

UserAuthen_LoginSuccess: User login succeeded.

UserAuthen_CombinedSuccess: The user is an iHealth user as well, measurement via SDK has been activated, and the data from the measurement belongs to the user.

UserAuthen_TrySuccess: Testing without internet connection succeeded.

UserAuthen_InvalidateUserInfo: UserID/clientID/clientSecret verification failed.

UserAuthen_SDKInvalidateRight: SDK has not been authorized.

UserAuthen_UserInvalidateRight: User has not been authorized.

UserAuthen_InternetError: Internet error, verification failed.

The measurement via SDK will be operated in the case of 1-4, and will be terminated if any of 5-8 occurs. The interface needs to be re-called after analyzing the return parameters.

Notice: when a new user registers via SDK, an 'iHealth disclaimer' will pop up automatically, and will require the user to agree in order to continue. SDK applications require an Internet connection; there is 10-day trial period if the SDK cannot connect to the internet, the SDK is fully functional during tryout period, but will be terminated without a working internet connection after 10 days.

disposeSynchronousTimeFinishBlock: Sync completed. Yes = Success, No = Fail.

disposeErrorBlock: Communication error codes, see section 5

(2) Real-time measurements

```
-(void)commandStartPO3MeasureData:(StartPO3MeasureData)startPO3MeasureData  
Measure:(DisposePO3MeasureData)disposePO3MeasureData  
FinishPO3MeasureData:(FinishPO3MeasureData)finishPO3MeasureData  
DisposeErrorBlock:(DisposePO3ErrorBlock)disposeErrorBlock;
```

Return parameters:

startPO3MeasureData: Start measurement. Return no for fail, return yes for success.

disposePO3MeasureData: SpO2 values, including SpO2, pulse rate, pulse intensity. Corresponding keys are spo2, bpm, wave, and pi.

finishPO3MeasureData: Finish measurement. No for fail, yes for success.

disposeErrorBlock: Communication error codes, see section 5

(3) Historical data

```
-(void)commandDisposePO3DataCount:(DisposePO3DataCount)disposePO3DataCount  
TransferMemoryData:(StartPO3Transmission)startTransmission  
Memory:(DisposePO3HistoryData)disposePO3HistoryData  
DisposePO3WaveHistoryData:(DisposePO3WaveHistoryData)disposePO3WaveHistoryData  
DisposeProgress:(DisposePO3ProgressData)progress  
FinishTransmission:(FinishPO3Transmission)finishTransmission  
DisposeErrorBlock:(DisposePO3ErrorBlock)disposeErrorBlock;
```

Return parameters:

disposePO3DataCount: Number of historical offline data measurements

startTransmission: Start data transmission. Yes for success, no for fail.

disposePO3HistoryData: date, spo2, bpm, and wave.

disposePO3WaveHistoryData: Pulse intensity, corresponding key: wave

progress: Data transmission progress from 0-1.0

finishTransmission: End transmission of data, yes for success, no for fail

disposeErrorBlock: Communication error codes, see section 5

(4) Restore factory settings

```
-(void)commandResetPO3DeviceDisposeResultBlock:(DisposePO3Block)disposeBlock  
DisposeErrorBlock:(DisposePO3ErrorBlock)disposeErrorBlock;
```

Return parameters:

disposeBlock: Returns yes for success, no for fail.

disposeErrorBlock: Communication error codes, see section 5

(5) Query power status

```
-(void)commandQueryBatteryInfo:(DisposePO3Block)disposeBlock  
DisposeErrorBlock:(DisposePO3ErrorBlock)disposeErrorBlock  
DisposeBattery:(DisposePO3Battery)disposeBattery;
```

Return parameters:

disposeBlock: Yes = success, no = fail.

disposeErrorBlock: Communication error codes, see section 5

disposeBattery: Battery %

(6) Disconnect connection

```
-(void)commandEndPO3CurrentConnect:(DisposePO3Block)disposeBlock  
DisposeErrorBlock:(DisposePO3ErrorBlock)disposeErrorBlock;
```

Return parameters:

disposeBlock: Yes = success, no = fail

disposeErrorBlock: Communication error codes, see section 5

4. Additional information

Device connection info: PO3ConnectNoti

Device disconnection info: PO3DisConnectNoti

When connecting multiple devices, distinguish between them via the serialNumber attribute.

5. Error codes

```
typedef enum{  
    PO3CommError = 0,    // Bluetooth Communication Error  
    PO3AccessError,      // Flash (Data) Access Error  
    PO3HardwareError,     // Irregular Hardware Error  
    PO3PRbpmtestError,    // The SpO2 or pulse rate test result is beyond the measurement range  
of the system  
    PO3UnknownError,     // Unknown Interference Detected  
    PO3SendCommandFaild, // Send failed  
    PO3DeviceDisConect,  // Device is disconnected  
    PO3DataZero,         // No data  
    PO3UserInvalidate     // User authentication fails  
}PO3ErrorID;
```

6. demo