Development Guide

1 Environment configuration

1.1 Add framework

Add `QCBandSDk.framework` to the project, the framework supports iOS 9.0 and above

Note: Because the classification is used in the framework, you need to add settings to the project

Target->Build Settings -> Other Linker Flags add **-ObjC**

1.2 Configure Bluetooth permissions

Configure bluetooth permissions in info.plist file

```
<key>NSBluetoothAlwaysUsageDescription</key>
<string>App needs to use your bluetooth device</string>
<key>NSBluetoothPeripheralUsageDescription</key>
<string>App needs to use your bluetooth device</string>
```

2 Usage

2.1 Service UUIDs supported by the device

在`QCSDKManager.h`中定义了，设备支持的服务UUID：

```
extern NSString *const QCBANDSDKSERVERUUID1;
extern NSString *const QCBANDSDKSERVERUUID2;
```

2.2 import framework library

Introduce the framework library into the code

```
#import <QCBandSDK/QCSDKManager.h>
#import <QCBandSDK/QCSDKCmdCreator.h>
```

Initialize `[QCSDKManager shareInstance]` with a singleton

> `QCSDKManager`:Peripherals for joining connections
>
> `QCSDKCmdCreator`:Used to send commands to peripherals

2.3 Scan Watch

initialization

Scanning can only be started when permissions are allowed and Bluetooth is turned on.

Import Apple's CoreBluetooth library and follow two protocols `<CBCentralManagerDelegate, CBPeripheralDelegate>`

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

Declare central and peripheral roles

```
/*Central Role,app*/
@property (strong, nonatomic) CBCentralManager *centerManager;

/*Peripheral role, scanned peripherals*/
@property (strong, nonatomic) NSMutableArray<CBPeripheral *> *peripherals;

/*Connected peripheral role*/
@property (strong, nonatomic) CBPeripheral *connectedPeripheral;
```

Instantiate the central role

```
self.centerManager = [[CBCentralManager alloc] initWithDelegate:self queue:nil];
```

Scan Watch

Using Scan Peripherals

```
NSArray *serviceUUIDStrings = @[QCBANDSDKSERVERUUID1,QCBANDSDKSERVERUUID2];

NSMutableArray *uuids = [NSMutableArray array];
for (id obj in serviceUUIDStrings) {
    if ([obj isKindOfClass:[NSString class]]) {
        CBUUID *uuid = [CBUUID UUIDWithString:obj];
        [uuids addObject:uuid];
    }
}

NSDictionary *option = @{CBCentralManagerScanOptionAllowDuplicatesKey : [NSNumber
numberWithBool:NO]};
[self.centerManager scanForPeripheralsWithServices:uuids options:option];
```

Note: To obtain the scanned peripheral devices in the agent, you can perform secondary filtering through the device name and other related information.

```
- (void)centralManager:(CBCentralManager *)central didDiscoverPeripheral:(CBPeripheral
*)peripheral advertisementData:(NSDictionary<NSString *,id> *)advertisementData RSSI:
(NSNumber *)RSSI {
    if (peripheral.name.length > 0) {
        [self.peripherals addObject:peripheral];
        [self.deviceList reloadData];
    }
}
```

2.4 Cancel the scan of the watch

Call the interface of the central role to stop scanning

```
[self.centerManager stopScan];
```

2.5 Connect the watch

start connecting

```
self.connectedPeripheral = self.peripherals[indexPath.row];
if (@available(iOS 13.0, *)) {
    [self.centerManager connectPeripheral:self.connectedPeripheral
options:@{CBConnectPeripheralOptionRequiresANCS : @YES}];
} else {
    [self.centerManager connectPeripheral:self.connectedPeripheral options:nil];
}
```

After the connection is successful, pass in the peripheral device to the SDK

```
- (void)centralManager:(CBCentralManager *)central didConnectPeripheral:(CBPeripheral
*)peripheral {
    [[QCSDKManager shareInstance] addPeripheral:peripheral];
}
```

2.6 Disconnect

```
[self.centerManager cancelPeripheralConnection:self.connectedPeripheral];
```

After disconnecting, remove peripherals

```
- (void)centralManager:(CBCentralManager *)central didDisconnectPeripheral:
(CBPeripheral *)peripheral error:(nullable NSError *)error {
    [[QCSDKManager shareInstance] removePeripheral:peripheral];
}
```

3. Instructions supported by the device

3.1 Set watch time

```
/**
 Set the time of the watch
 */
+ (void)setTime:(NSDate *)date success:(void (^)(NSDictionary *))suc failed:(void (^)
(void))fail;
```

3.2 Read watch battery

```
/*!
 *  @func Read watch battery
 *  @param suc battery:Power level(0~8)
 */
+ (void)readBatterySuccess:(void (^)(int battery))suc failed:(void (^)(void))fail;
```

3.3 Bound Vibration

```
/**
 *  Bound Vibration
 */
+ (void)alertBindingSuccess:(nullable void (^)(void))suc fail:(nullable void (^)
(void))fail;
```

3.4 Set wristband time base/user personal information

```
/**
 Set wristband time base/user personal information

 @param twentyfourHourFormat    : YES 24 hour system; NO 12 hour system
 @param metricSystem            : YES Metric; NO Imperial
 @param gender                  : gender (0=male, 1=female)
 @param age                     : age (years)
 @param height                  : height (cm)
 @param weight                  : weight (kg)
 @param sbpBase                 : systolic blood pressure base (mmhg) (reserved value,
Defaults:0)
 @param dbpBase                 : Diastolic blood pressure base (mmhg) (reserved value,
Defaults:0)
 @param hrAlarmValue            : Heart rate alarm value (bpm) (reserved value,
Defaults:0)
 *
 *
 */
+ (void)setTimeFormatTwentyfourHourFormat:(BOOL)twentyfourHourFormat
    metricSystem:(BOOL)metricSystem
    gender:(NSInteger)gender
    age:(NSInteger)age
    height:(NSInteger)height
    weight:(NSInteger)weight
    sbpBase:(NSInteger)sbpBase
    dbpBase:(NSInteger)dbpBase
    hrAlarmValue:(NSInteger)hrAlarmValue
    success:(void (^)(BOOL, BOOL, NSInteger, NSInteger, NSInteger, NSInteger,
NSInteger, NSInteger, NSInteger))success
    fail:(void (^)(void))fail;
```

3.5 Get watch time base/user personal information

```
/**
 Get watch time base/user personal information

 @param success isTwentyfour: YES 24 hour system; NO 12 hour system
                isMetricSystem: YES Metric; NO Imperial
                gender: gender (0=male, 1=female)
                age: age (years)
                height: height (cm)
                weight: weight (kg)
                sbpBase: systolic blood pressure base (mmhg) (reserved value,
Defaults:0)
                dbpBase: Diastolic blood pressure base (mmhg) (reserved value,
Defaults:0)
                hrAlarmValue: Heart rate alarm value (bpm) (reserved value, Defaults:0)
 */
+ (void)getTimeFormatInfo:(nullable void (^)(BOOL isTwentyfour, BOOL isMetricSystem,
NSInteger gender, NSInteger age, NSInteger height, NSInteger weight, NSInteger sbpBase,
NSInteger dbpBase, NSInteger hrAlarmValue))success fail:(nullable void (^)(void))fail;
```

3.6 Get the version number of the watch firmware

```
/**
 *  @func Get the version number of the watch firmware
 *
 *  @param success The format of software and hardware version numbers is
generally"x.x.x"
 */
+ (void)getDeviceSoftAndHardVersionSuccess:(void (^)(NSString *_Nonnull, NSString
*_Nonnull))success fail:(void (^)(void))fail;
```

3.7 Find receive push messages

```
/*!
 *  @func Find receive push messages
 *
 *  @param  suc filters :Support message  0:telephone 1:SMS; 2:QQ 3:wechat 4:FaceBook
5:WhatsApp 6:twitter 7:skype 8:line 9:linkedin 10:instagram 11:tim 12:snapchat
13:reserved 14:reserved 15:other
 *          Fx -
>@[@"1",@"0",@"0",@"0",@"0",@"0",@"0",@"0",@"0",@"0",@"0",@"0",@"0",@"0",@"0"]
Indicates receiving incoming call alerts
 */
+ (void)getFilterSuccess:(void (^)(NSArray<NSNumber *> *filters))suc failed:(void (^)
(void))fail;
```

3.8 Set up to receive push messages

```
/*!
 *  @func Set up to receive push messages
 *
 *  @param  filters :Support message  0:telephone 1:SMS; 2:QQ 3:wechat 4:FaceBook
5:WhatsApp 6:twitter 7:skype 8:line 9:linkedin 10:instagram 11:tim 12:snapchat
13:reserved 14:reserved 15:other
 *          Fx -
>@[@"1",@"0",@"0",@"0",@"0",@"0",@"0",@"0",@"0",@"0",@"0",@"0",@"0",@"0",@"0"]
Indicates receiving incoming call alerts
 */
+ (void)setFilter:(NSArray *)filters success:(void (^)(void))suc failed:(void (^)
(void))fail;
```

3.9 ~~Read the storage distribution of the watch~~ （Deprecated）

```
/*!
 *  @func Read the storage distribution of the watch (get which day the watch has
stored pedometer data)
 *  @discussion FX  ->@[@(0),@(4)], 0 means today, Data available 4 days ago
 */
+ (void)getSaveDataIndex:(void (^)(NSArray *indexs))suc failed:(void (^)(void))fail;
```

3.10 Get current steps

```
/*!
 *  @func   Get current steps
 */
+ (void)getCurrentSportSucess:(void (^)(SportModel *sport))suc failed:(void (^)
(void))fail;
```

3.11 Get total statistics for a day(Steps、Calories、Distance、Time)

```
/*!
 *  @func   Get total statistics for a day
 *
 *  @param   index  : 0->Today 1->1 day ago.Maximum 29
 *  @param   suc    :Using this command cannot accurately obtain the statistical data
of the day, the device will save the data every 15 minutes, so there will be a 15-
minute interval
 */
+ (void)getOneDaySportBy:(NSInteger)index success:(void (^)(SportModel *model))suc
fail:(void (^)(void))fail;
```

3.12 Get detailed exercise data for a day

```
/*!
 *  @func   Get detailed exercise data for a day
 *  @discussion There is a tick every 15 minutes, and there will be a maximum of 96
pieces of data per day. For details, please see the returned content
 *  @param items  sports:return all sports models
 */
+ (void)getSportDetailDataByDay:(NSInteger)dayIndex sportDatas:(nullable void (^)
(NSArray<SportModel *> *sports))items fail:(nullable void (^)(void))fail;
```

3.13 Get detailed exercise data for a specified time period on a certain day

```
/*!
 *  @func   Get detailed exercise data for a specified time period on a certain day
 *  @param  minuteInterval  minute interval for each index
 *  @param  beginIndex      time period start index
 *  @param  endIndex        time period end index
 *  @param  items           sports:return all sports models
 */
+ (void)getSportDetailDataByDay:(NSInteger)dayIndex minuteInterval:
(NSInteger)minuteInterval beginIndex:(NSInteger)beginIndex endIndex:(NSInteger)endIndex
sportDatas:(nullable void (^)(NSArray<SportModel *> *sports))items fail:(nullable void
(^)(void))fail;
```

3.14 Get detailed sleep data for a day

```
//sleep data type
typedef NS_ENUM(NSInteger, SLEEPTYPE) {
    SLEEPTYPENONE,     //no data
    SLEEPTYPESOBER,    //wide awake
    SLEEPTYPELIGHT,    //light sleep
    SLEEPTYPEDEEP,     //Deep sleep
    SLEEPTYPEUNWEARED //not worn
};


@interface QCSleepModel : NSObject
@property (nonatomic, assign) SLEEPTYPE type;       //sleep type
@property (nonatomic, strong) NSString *happenDate; //Start Time yyyy-MM-dd HH:mm:ss
@property (nonatomic, strong) NSString *endTime;    //End Time yyyy-MM-dd HH:mm:ss.
@property (nonatomic, assign) NSInteger total;      //Time interval between start time
and end time (unit: minutes)
@end


/*!
 *  @func  Get detailed sleep data for a day
 *  @discussion The time period corresponding to each sleep type, please see the
returned content for details
 *  @param items    sleeps:return all sleep models
 */
+ (void)getSleepDetailDataByDay:(NSInteger)dayIndex sleepDatas:(nullable void (^)
(NSArray<QCSleepModel *> *sleeps))items fail:(nullable void (^)(void))fail;
```

3.15 ~~Get detailed sleep data for a specified time period of a day~~（Deprecated）

```
/*!
 *  @func   Get detailed sleep data for a specified time period of a day(Deprecated)
 *  @param  minuteInterval  minute interval for each index
 *  @param  beginIndex      time period start index
 *  @param  endIndex        time period end index
 *  @param  items           sports:return all sleep models
 */
+ (void)getSleepDetailDataByDay:(NSInteger)dayIndex minuteInterval:
(NSInteger)minuteInterval beginIndex:(NSInteger)beginIndex endIndex:(NSInteger)endIndex
sleepDatas:(nullable void (^)(NSArray<SleepModel *> *sleeps))items fail:(nullable void
(^)(void))fail;
```

3.16 Get sedentary reminders

```
/**
 * Get sedentary reminders
 * @note beginTime: Start time(Formart"HH:mm")
 * @note endTime: End time(Formart"HH:mm")
 * @note repeat: Reminder repeat cycle(Array order is Sunday-Saturday, Such as @[@0,
@1, @0, @0, @0, @0, @0], means repeat every Monday)
 * @note interval: Reminder interval/period (unit: minutes, range: 1-255)
 */
+ (void)getSitLongRemindResult:(void (^)(NSString *beginTime, NSString *endTime,
NSArray *repeat, NSUInteger interval))remind fail:(void (^)(void))fail;
```

3.17 Set sedentary reminders

```
/**
 * Set sedentary reminders
 * @param beginTime: Start time(Formart"HH:mm")
 * @param endTime: End time(Formart"HH:mm")
 * @param repeat: Reminder repeat cycle(Array order is Sunday-Saturday, Such as @[@0,
@1, @0, @0, @0, @0, @0], means repeat every Monday)
 * @param interval: Reminder interval/period (unit: minutes, range: 1-255)
 */
+ (void)setBeginTime:(NSString *)beginTime endTime:(NSString *)endTime repeatModel:
(NSArray *)repeat timeInterval:(NSUInteger)interval success:(void (^)(void))suc fail:
(void (^)(void))fail;
```

3.18 Find Watch

```
/**
 * Find Watch
 */
+ (void)lookupDeviceSuccess:(void (^)(void))suc fail:(void (^)(void))fail;
```

3.19 ~~Start heart rate measurement~~ （Deprecated）

```
/*!
 *  @func Start heart rate measurement
 */
+ (void)startHeartRateMeasuringWithSuccess:(nullable void (^)(void))suc fail:(nullable
void (^)(void))fail;
```

3.20 ~~End heart rate measurement~~ （Deprecated）

```
/*!
 *  @func End heart rate measurement
 *  @param hr: The measured heart rate value (the watch needs to display the
measurement result based on this value)
 */
+ (void)endHeartRateMeasuringWithHR:(NSInteger)hr success:(nullable void (^)(void))suc
fail:(nullable void (^)(void))fail;
```

3.21 ~~Start blood oxygen measurement~~ （Deprecated）

```
/*!
 *  @func Start blood oxygen measurement
 */
+ (void)startBloodOxygenMeasuringWithSuccess:(nullable void (^)(void))suc fail:
(nullable void (^)(void))fail;
```

3.22 ~~End blood oxygen measurement~~ （Deprecated）

```
/*!
 *  @func End blood oxygen measurement
 *  @param soa2: The measured blood oxygen value (the watch needs to display the
measurement result based on this value)
 */
+ (void)endBloodOxygenMeasuringWithSoa2:(CGFloat)soa2 success:(nullable void (^)
(void))suc fail:(nullable void (^)(void))fail;
```

3.23 ~~Start blood pressure measurement~~（Deprecated）

```
/*!
 *  @func Start blood pressure measurement
 */
+ (void)startBloodPressureMeasuringSuccess:(nullable void (^)(void))suc fail:(nullable
void (^)(void))fail;
```

3.24 ~~End blood pressure measurement~~（Deprecated）

```
/*!
 *  @func End blood pressure measurement
 *  @param sbp:  Measured systolic blood pressure value (the watch needs to display the
measurement result based on this value)
 *  @param dbp:  Measured diastolic blood pressure value (the watch needs to display
the measurement result based on this value)
 */
+ (void)endBloodPressureMeasuringWithSbp:(NSInteger)sbp dbp:(NSInteger)dbp success:
(nullable void (^)(void))suc fail:(nullable void (^)(void))fail;
```

3.25 ~~Turn on the switch of one-key physical examination measurement~~（Deprecated）

```
/*!
 *  @func Turn on the switch of one-key physical examination measurement
 */
+ (void)openOneKeyExaminationSwitchWithSampleRate:(NSInteger)rate success:(nullable
void (^)(void))suc fail:(nullable void (^)(void))fail;
```

3.26 ~~Turn off the switch of one-key physical examination measurement~~ （Deprecated）

```
/*!
 *  @func Turn off the switch of one-key physical examination measurement
 */
+ (void)closeOneKeyExaminationSwitchSuccess:(nullable void (^)(void))suc fail:(nullable
void (^)(void))fail;
```

3.27 Set a reminder to drink water

```
/*!
 *  @func Set a reminder to drink water
 *  @param index: reminder number
 *  @param time: Format: HH:mm
 *  @param cycle: The time is Sunday-Saturday, the format: such as @[@0, @1, @0, @0,
@0, @0, @0], it means repeat every Monday
 */
+ (void)setDrinkWaterRemindIndex:(NSUInteger)index type:(ALARMTYPE)type time:(NSString
*)time cycle:(NSArray<NSNumber *> *)cycle success:(nullable void (^)(void))suc failed:
(nullable void (^)(void))fail;
```

3.28 Get water reminders

```
/**
 * @func Get water reminders
 * @param index: reminder number
 * @note type: Alarm type
 * @note time: Time, the format is "HH:mm"
 * @note cycle: Reminder period, the array order is Sunday-Saturday, such as @[@0, @1,
@0, @0, @0, @0, @0], it means repeat every Monday
 */
+ (void)getDrinkWaterRemindWithIndex:(NSUInteger)index remind:(nullable void (^)
(NSUInteger index, ALARMTYPE type, NSString *time, NSArray<NSNumber *> *cycle))remind
fail:(nullable void (^)(void))fail;
```

3.29 Get information about the wrist flip feature

```
/**
 *  @func Get information about the wrist flip feature
 *  @param success isOn: Whether the function is enabled; leftHandWear: Whether to wear
on the left hand
 */
+ (void)getFlipWristInfo:(nullable void (^)(BOOL isOn, NSUInteger flipType))success
fail:(void (^)(void))fail;
```

3.30 Information about setting the wrist flip function

```
/**
 *  Information about setting the wrist flip function
 *  @param on:       Whether the function is enabled
 *  @param flipType: Whether to wear on the left hand
 */
+ (void)setFlipWristOn:(BOOL)on flipType:(NSUInteger)flipType success:(nullable void
(^)(BOOL featureOn, NSUInteger flipType))success fail:(nullable void (^)(void))fail;
```

3.31 Get information about the Do Not Disturb feature

```
/**
 *  @func Get information about the Do Not Disturb feature
 *  @param success isOn: Whether the function is enabled; begin: Starting time, Format
HH: mm; end: End Time, Format HH: mm
 */
+ (void)getDontDisturbInfo:(nullable void (^)(BOOL isOn, NSString *begin, NSString
*end))success fail:(void (^)(void))fail;
```

3.32 Information on setting up the Do Not Disturb feature

```
/**
 *  Information on setting up the Do Not Disturb feature
 *  @param on       Whether the function is enabled
 *  @param begin    Starting time, Format HH: mm
 *  @param end      End Time, Format HH: mm
 */
+ (void)setDontDisturbOn:(BOOL)on beginTime:(NSString *)begin endTime:(NSString *)end
success:(nullable void (^)(BOOL featureOn, NSString *begin, NSString *end))success
fail:(nullable void (^)(void))fail;
```

3.33 Switch to the camera interface

```
/**
 *  Switch the lower computer (watch) to the camera interface
 */
+ (void)switchToPhotoUISuccess:(nullable void (^)(void))success fail:(nullable void (^)
(void))fail;
```

3.34 keep the camera interface

```
/**
 *  Keep the camera interface of the lower computer (watch)
 */
+ (void)holdPhotoUISuccess:(nullable void (^)(void))success fail:(nullable void (^)
(void))fail;
```

3.35 Stop taking pictures

```
/**
 *  Stop the lower computer (watch) to take pictures
 */
+ (void)stopTakingPhotoSuccess:(nullable void (^)(void))success fail:(nullable void (^)
(void))fail;
```

3.36 Restart the watch

```
/**
 Restart the watch
 */
+ (void)resetBandHardlySuccess:(nullable void (^)(void))suc fail:(nullable void (^)
(void))fail;
```

3.37 Get watch Mac address

```
/**
 *  @func Get watch Mac address
 *  @param success The Mac address format is "AA:BB:CC:DD:EE:FF"
 */
+ (void)getDeviceMacAddressSuccess:(nullable void (^)(NSString *_Nullable
macAddress))success fail:(nullable void (^)(void))fail;
```

3.38 Get information about the timed blood pressure measurement function

```
/**
 *  Get information about the timed blood pressure measurement function
 *  @param success  featureOn       YES: ON; NO: OFF
 *                  beginTime       Start time, format "HH:mm"
 *                  endTime         End time, the format is "HH:mm"
 *                  minuteInterval  minute interval (minutes)
 */
+ (void)getSchedualBPInfo:(nullable void (^)(BOOL featureOn, NSString *beginTime,
NSString *endTime, NSInteger minuteInterval))success fail:(void (^)(void))fail;
```

3.39 Information on setting the timed blood pressure measurement function

```
/**
 *  Information on setting the timed blood pressure measurement function
 *  @param featureOn        YES: ON; NO: OFF
 *  @param beginTime        Start time, format "HH:mm"
 *  @param endTime          End time, the format is "HH:mm"
 *  @param minuteInterval   minute interval (minutes)
 */
+ (void)setSchedualBPInfoOn:(BOOL)featureOn beginTime:(NSString *)beginTime endTime:
(NSString *)endTime minuteInterval:(NSInteger)minuteInterval success:(nullable void (^)
(BOOL featureOn, NSString *beginTime, NSString *endTime, NSInteger
minuteInterval))success fail:(void (^)(void))fail;
```

3.40 Obtain historical data for timed blood pressure measurements

```
/**
 *  Obtain historical data for timed blood pressure measurements
 *  @param userAge  :User age
 *  @param success  data: Heart rate module data, the current reply is actually unified as heart rate, which can be processed by itself in the callback
 */
+ (void)getSchedualBPHistoryDataWithUserAge:(NSInteger)userAge success:(nullable void (^)(NSArray<BloodPressureModel *> *data))success fail:(nullable void (^)(void))fail;
```

3.41 Reset the watch to factory settings

```
/**
 *  Reset the watch to factory settings
 */
+ (void)resetBandToFacotrySuccess:(nullable void (^)(void))success fail:(nullable void (^)(void))fail;
```

3.42 Get historical data of exercise records

```
/**
 * @func Get historical data of exercise records
 * @param lastUnixSeconds The time when the last exercise data occurred (seconds since 1970-01-01 00:00:00)
 * @note success models Motion data array
 */
+ (void)getExerciseDataWithLastUnixSeconds:(NSUInteger)lastUnixSeconds getData:(nullable void (^)(NSArray<ExerciseModel *> *models))getData fail:(nullable void (^)(void))fail;
```

3.43 Get historical data for manual blood pressure measurements

```
/**
 *  Get historical data for manual blood pressure measurements
 *  @param lastUnixSeconds Time when the last manual blood pressure data occurred (seconds since 1970-01-01 00:00:00)
 *  @param success  data blood pressure data array
 */
+ (void)getManualBloodPressureDataWithLastUnixSeconds:(NSUInteger)lastUnixSeconds success:(nullable void (^)(NSArray<BloodPressureModel *> *data))success fail:(nullable void (^)(void))fail;
```

3.44 Get timed heart rate historical data

```
/**
 * @func Get timed heart rate historical data
 * @param dates: List of dates for which historical data needs to be obtained
 * @note success models Timed heart rate data array
 */
+ (void)getSchedualHeartRateDataWithDates:(NSArray<NSDate *> *)dates success:(nullable
void (^)(NSArray<SchedualHeartRateModel *> *models))success fail:(nullable void (^)
(void))fail;


/**
 * @func Get timed heart rate historical data
 * @param dayIndexs The number of days for which historical data needs to be obtained
(0->today, 1->yesterday, 2->the day before yesterday, and so on)
 * @note success models Timed heart rate data array
 */
+ (void)getSchedualHeartRateDataWithDayIndexs:(NSArray<NSNumber*> *)dayIndexs success:
(void (^)(NSArray<QCSchedualHeartRateModel *> *_Nonnull))success fail:(void (^)
(void))fail;
```

3.45 Get information about the timed heart rate function

```
/**
 *  Get information about the timed heart rate function
 *  @param success  enable  Whether the timed heart rate function is enabled. YES: ON;
NO: OFF
 */
+ (void)getSchedualHeartRateStatusWithCurrentState:(BOOL)enable success:(nullable void
(^)(BOOL enable))success fail:(nullable void (^)(void))fail;
```

3.46 Information on setting the timed heart rate function

```
/**
 *  Information on setting the timed heart rate function
 *  @param enable   Whether the timed heart rate function is enabled. YES: ON; NO: OFF
 */
+ (void)setSchedualHeartRateStatus:(BOOL)enable success:(nullable void (^)(BOOL
enable))success fail:(nullable void (^)(void))fail;
```

3.47 Get information about the weather forecast feature

```
/**
 *  Get information about the weather forecast feature
 *
 *  @param enable:  Whether the weather forecast function is enabled(YES: ON; NO: OFF).
 *  @param temperatureUsingCelsius: whether to use degrees Celsius(YES: ON; NO: OFF).
 *  @param success: success callback
 *  @param success: fail callback
 */
+ (void)getWeatherForecastStatusWithCurrentState:(BOOL)enable temperatureUsingCelsius:
(BOOL)temperatureUsingCelsius success:(nullable void (^)(BOOL enable, BOOL
usingCelsius))success fail:(nullable void (^)(void))fail;
```

3.48 Set the information of the weather forecast function

```
/**
 *  Set the information of the weather forecast function
 *
 *
 *  @param enable:  Whether the weather forecast function is enabled(YES: ON; NO: OFF).
 *  @param temperatureUsingCelsius: whether to use degrees Celsius(YES: ON; NO: OFF).
 *  @param success: success callback
 *  @param success: fail callback
 */
+ (void)setWeatherForecastStatus:(BOOL)enable temperatureUsingCelsius:
(BOOL)temperatureUsingCelsius success:(nullable void (^)(BOOL enable, BOOL
usingCelsius))success fail:(nullable void (^)(void))fail;
```

3.49 Send the content of the weather forecast to the watch

```
/**
 Send the content of the weather forecast to the watch
 *
 * @param contents:   [{"time":Date timestamp (note that it needs to be modified to the
current time zone),
                        "type":weather type,
                        "low-temp":temperature minimum,
                        "high-temp":temperature maximum,
                        "humidity":Humidity value,
                        "needUmbrella":whether to bring an umbrella}]
 *
 *
 *
 * @describe weather type:  0=unknown,
                           1=sunny,
                           2=partly cloudy,
                           3 =rain,
                           4=Snow,
                           5=smog,
                           6=thunderbolt
 */
+ (void)sendWeatherContents:(NSArray<NSDictionary *> *)contents success:(nullable void
(^)(void))success fail:(nullable void (^)(void))fail;
```

3.50 Get information about the watch's brightness adjustment function

```
/**
 Get information about the watch's brightness adjustment function
 @param success lightLevel Brightness level, 1 - 10 => 10% - 100%
 */
+ (void)getDeviceLightLevelWithCurrentLevel:(NSInteger)lightLevel success:(nullable
void (^)(NSInteger lightLevel))success fail:(nullable void (^)(void))fail;
```

3.51 Information on setting the device's brightness adjustment function

```
/**
 Information on setting the device's brightness adjustment function
 @param lightLevel Brightness level, 1 - 10 => 10% - 100%
 */
+ (void)setDeviceLightLevel:(NSInteger)lightLevel success:(nullable void (^)(NSInteger
lightLevel))success fail:(nullable void (^)(void))fail;
```

3.52 Get/set the bracelet display time & user homepage parameter function information

```
/**
 *  Get/set the bracelet display time & user homepage parameter function information
 *  @param opType 0X01=read, 0x02=Write , 0x03=The homepage picture is restored to
default (AA=0x03 BBCCDDEE invalid)
 *  @param lightingSeconds Bright screen time, Unit: second. The legal value is 4~10
(second).
 *  @param homePageType Home page optional display data type (0=invalid, 1=steps,
2=calories, 3=weather, 4=heart rate)
```

```
 *  @param transparency The mask transparency setting of the home page (0~100, 0=the
mask is opaque/the base image is not displayed, 100=the mask is fully transparent/the
base image is displayed)
 *  @param pictureType 0=default homepage picture, 1=user-configured homepage picture
(only for reading, invalid when writing)
 */
+ (void)setHomePageScreenOpType:(NSInteger)opType lightingSeconds:
(NSInteger)lightingSeconds homePageType:(NSInteger)homePageType transparency:
(NSInteger)transparency pictureType:(NSInteger)pictureType success:(nullable void (^)
(NSInteger lightingSeconds, NSInteger homePageType, NSInteger transparency, NSInteger
pictureType))suc fail:(nullable void (^)(void))fail;

/**
 *  Set the information on the display time of the bracelet
 *  @param seconds Screen-on time, unit: second. The legal value is 4~10 (seconds).
 */
+ (void)setLightingSeconds:(NSInteger)seconds success:(nullable void (^)(void))suc
fail:(nullable void (^)(void))fail;

/**
 *  Get information about display duration
 *  @param suc lightingSeconds Screen-on time, unit: second. The legal value is 4~10
(seconds).
 */
+ (void)getLightingSecondsWithSuccess:(nullable void (^)(NSInteger seconds))suc fail:
(nullable void (^)(void))fail;
```

3.53 According to the specified time stamp, the new version of Sports+ (V2) data summary information

```
/**
 According to the specified time stamp, the new version of Sports+ (V2) data summary
information
 @param timestamp
 @param finished spSummary - Motion+Summary info array
 */
+ (void)getSportPlusSummaryFromTimestamp:(NSTimeInterval)timestamp finished:(nullable
void (^)(NSArray *_Nullable spSummary, NSError *_Nullable error))finished;
```

3.54 According to the specified new version of the campaign + summary information, get some summary information and detailed data of the campaign

```
/**
 According to the specified new version of the campaign + summary information, get some
summary information and detailed data of the campaign
 @param finished spSummary - Motion+Summary info array
 */
+ (void)getSportPlusDetailsWithSummary:(OdmGeneralExerciseSummaryModel *)summary
finished:(nullable void (^)(OdmGeneralExerciseSummaryModel *_Nullable summary,
OdmGeneralExerciseDetailModel *_Nullable detail, NSError *_Nullable error))finished;
```

3.55 Get a list of file requirements files

```
/**
 Get a list of file requirements files
 */
+ (void)getNeededFileListFinished:(nullable void (^)(NSArray<NSString *> *_Nullable
fileList, NSError *_Nullable error))finished;
```

3.56 Get/set user target information

```
/**
 Get/set user target information

 * @param suc stepTarget:     Step target
           calorieTarget:     Calorie Goal, Unit: Calories
          distanceTarget:     Distance to target, unit: meters
           sportDuration:     Exercise duration target Unit: minutes (reserved value,
default: 0)
            sleepDuration:    Sleep duration target unit: minutes (reserved value,
default: 0)
 */
+ (void)getStepTargetInfoWithSuccess:(nullable void (^)(NSInteger stepTarget,NSInteger
calorieTarget,NSInteger distanceTarget,NSInteger sportDuration,NSInteger
sleepDuration))suc fail:(nullable void (^)(void))fail;

/**
 Set user target information

 * @param stepTarget:       Step target
 * @param calorieTarget:    Calorie Goal, Unit: Calories
 * @param distanceTarget:   Distance to target, unit: meters
 * @param sportDuration:    Exercise duration target Unit: minutes (reserved value,
default: 0)
 * @param sleepDuration:    Sleep duration target unit: minutes (reserved value,
default: 0)
 */
+ (void)setStepTarget:(NSInteger)stepTarget calorieTarget:(NSInteger)calorieTarget
distanceTarget:(NSInteger)distanceTarget sportDurationTarget:(NSInteger)sportDuration
sleepDurationTarget:(NSInteger)sleepDuration success:(nullable void (^)(void))suc fail:
(nullable void (^)(void))fail;
```

3.57 Get a list of watch face files

```
/**
  Get a list of watch face files
*/
+ (void)listDialFileFinished:(nullable void (^)(NSArray <SimpleDialFileModel
*>*_Nullable dialFiles, NSError *_Nullable error))finished;
```

3.58 delete watch face file

```
/**
 *  delete watch face file
 */
+ (void)deleteDialFileName:(NSString *)fileName force:(BOOL)force finished:(nullable
void (^)(NSError *_Nullable error))finished;
```

3.59 Obtain historical data of timed body temperature measurement

```
/**
*  Obtain historical data of timed body temperature measurement
*/
+ (void)getSchedualTemperatureDataByDayIndex:(NSInteger)dayIndex finished:(nullable
void (^)(NSArray *_Nullable temperatureList, NSError *_Nullable error))finished;
```

3.60 Get historical data for manual body temperature measurements

```
/**
 *  Get historical data for manual body temperature measurements
 */
+ (void)getManualTemperatureDataByDayIndex:(NSInteger)dayIndex finished:(nullable void
(^)(NSArray *_Nullable temperatureList, NSError *_Nullable error))finished;
```

3.61 Get historical data for blood oxygen measurements

```
/**
 *  Get historical data for blood oxygen measurements
 */
+ (void)getBloodOxygenDataByDayIndex:(NSInteger)dayIndex finished:(void (^)(NSArray *
_Nullable, NSError * _Nullable))finished;
```

3.62 Get custom dial parameters

```
/**
 *  Get custom dial parameters
 */
+ (void)getDailParameterWithFinished:(void (^)(DialParameterModel * _Nullable time,
DialParameterModel * _Nullable date, DialParameterModel * _Nullable value, NSError *
_Nullable))finished;
```

3.63 Set custom watch face parameters

```
/**
 *  Set custom watch face parameters
 */
+ (void)setDailParameter:(DialParameterModel * _Nullable)time date:(DialParameterModel
* _Nullable)date value:(DialParameterModel * _Nullable)value finished:(void (^)
(DialParameterModel * _Nullable time, DialParameterModel * _Nullable date,
DialParameterModel * _Nullable value, NSError * _Nullable))finished;
```

3.64 Get watch alarm

```
/**
 *  Get watch alarm
 */
+ (void)getBandAlarmsWithFinish:(void(^)(NSArray <AlarmModel*>* _Nullable,NSError *
_Nullable))finished;
```

3.65 Set watch alarm

```
/**
 *  Set watch alarm
 */
+ (void)setBandAlarms:(NSArray <AlarmModel*>*)alarms finish:(void(^)(NSArray *
_Nullable,NSError * _Nullable))finished;
```

3.66 Menstrual period reminder function settings

```
/**
Menstrual period reminder function settings: send settings to the watch

 * @param open:             Switch 1=on, 0 off, 2=invalid (when the APP reads the
configuration from the wristband, if this bit is 2, the wristband parameter is invalid)
 * @param durationday:      Menstrual period duration, in days; (default 6 days)
 * @param intervalday:      Menstrual cycle, unit day; (default 28 days)
 * @param startday:         How many days ago was the last start, 0=starts today;
(default 0 means, APP does not display)
 * @param endday:           How many days ago was the last end, 0=end today; (the
default value of 0 means that the APP does not display) (when endday is not equal to
startday+durationday, it means that the user manually modified the end time)
 * @param remindOpen:       Reminder switch 1=on, others off; (default off)
 * @param beforemenstrday:  How many days in advance to remind the menstrual period, 1
= one day in advance. 1~3. (default 2)
 * @param beforeovulateday: How many days in advance to remind the ovulation period,
1~3. (default 2)
 * @param hour: Reminder time point-Hour
 * @param minute: Reminder time point - minutes
 */
+ (void)sendMenstrSettingFeatures:(BOOL)open
                      durationDay:(NSString*)durationday
                      intervalDay:(NSString*)intervalday
                         startDay:(NSString*)startday endDay:(NSString*)endday
                      remindState:(BOOL)remindOpen
                  menstrBeforeDay:(NSString*)beforemenstrday
```

```
                 ovulateBeforeDay:(NSString*)beforeovulateday
                       remindHour:(NSString*)hour
                     remindMinute:(NSString*)minute
                         finished:(void (^)(void))finished;
```

3.67 Send firmware file

```
/**
 Send the firmware file and request to use the bin file to upgrade, the result will be processed in the callback

 @param data              OTA binary character stream
 @param start             start sending callback
 @param percentage        progress callback
 @param success           success callback
 @param failed            failure callback
 */

+ (void)syncOtaBinData:(NSData *)data
                 start:(nullable void (^)(void))start
            percentage:(nullable void (^)(int percentage))percentage
               success:(nullable void (^)(int seconds))success
                failed:(nullable void (^)(NSError *error))failed;
```

3.68 Send watch face file

```
/**
 Send the watch face file, require the use of the bin file

 @param name              watch face file name
 @param data              Dial binary character stream
 @param start             start sending callback
 @param percentage        progress callback
 @param success           success callback
 @param failed            failure callback
 */
+ (void)syncDialFileName:(NSString *)name
                 binData:(NSData *)data
                   start:(nullable void (^)(void))start
              percentage:(nullable void (^)(int percentage))percentage
                 success:(nullable void (^)(int seconds))success
                  failed:(nullable void (^)(NSError *error))failed;
```

3.69 Send files missing from watch

```
/**
 Send files missing from watch

 @param name              Watch missing file name
 @param data              watch missing file binary character stream
 @param start             start sending callback
 @param percentage        progress callback
 @param success           success callback
 @param failed            failure callback
 */
+ (void)syncResourceFileName:(NSString *)name
                     binData:(NSData *)data
                       start:(nullable void (^)(void))start
                  percentage:(nullable void (^)(int percentage))percentage
                     success:(nullable void (^)(int seconds))success
                      failed:(nullable void (^)(NSError *error))failed;
```

3.70 Send a picture watch face file

```
/**
 Send the picture watch face file, and request the (pixel) size to be cropped to the
size of the current bracelet (the watch will verify the width and height of the
picture)

 @param img               watch face image
 @param start             start sending callback
 @param percentage        progress callback
 @param success           success callback
 @param failed            failure callback
 */
+ (void)syncImage:(UIImage *)img
            start:(nullable void (^)(void))start
       percentage:(nullable void (^)(int percentage))percentage
          success:(nullable void (^)(int seconds))success
           failed:(nullable void (^)(NSError *error))failed;
```

3.71 Receive a watch message

```
@interface QCSDKManager : NSObject

/*
 *  Receive notifications from watch, find phone
 */
@property(nonatomic,copy)void(^findPhone)(void);

/*
 *  Receive notifications from watch, enter camera
 */
@property(nonatomic,copy)void(^switchToPicture)(void);

/*
 *  Receive notification of watch, take photo
 */
@property(nonatomic,copy)void(^takePicture)(void);

/*
 *  Receive a notification from the watch to end taking pictures
 */
@property(nonatomic,copy)void(^stopTakePicture)(void);

// singleton class instance
+ (instancetype)shareInstance;

@end
```

3.72 Set/get timed blood oxygen switch status

```
/**
 *  Information on setting the timed oximetry function
 *  @param featureOn        YES: ON; NO: OFF
 */

+ (void)setSchedualBOInfoOn:(BOOL)featureOn success:(nullable void (^)(BOOL
featureOn))success fail:(void (^)(void))fail;

/**
 *  Get information about the timed oximetry function
 *  @param success featureOn YES: 开启; NO: 关闭
 */

+ (void)getSchedualBOInfoSuccess:(nullable void (^)(BOOL featureOn))success fail:(void
(^)(void))fail;
```

3.73 Send measurement commands (commands are encapsulated in QCSDKManager)

```
typedef NS_ENUM(NSInteger, QCMeasuringType) {
    QCMeasuringTypeHeartRate = 0,   //Heart rate measurement
    QCMeasuringTypeBloodPressue,    //blood pressure measurement
    QCMeasuringTypeBloodOxygen,     //blood oxygen measurement
    QCMeasuringTypeCount,
};

//The measurement result is the result in the hanle callback
//When measuring heart rate, result returns NSNumber: @(60)
//When measuring blood pressure, the result returned is
NSDictionary:@{@"sbp":@"120",@"dbp":@"60"}
//When measuring blood oxygen, the result returns NSNumber: @(98)

/// Send measurement order
/// @param type: Measurement type
/// @param handle: Measurement result callback (error code: -1: failed to send start
command, -2: failed to send end command, -3: bracelet is not properly worn)
- (void)startToMeasuringWithOperateType:(QCMeasuringType)type completedHandle:(void(^)
(BOOL isSuccess,id result,NSError *error))handle;



/// stop measurement command
/// @param type: Measurement type
/// @param handle: Measurement result callback (error code:-1: Failed to send end
command)
- (void)stopToMeasuringWithOperateType:(QCMeasuringType)type completedHandle:(void(^)
(BOOL isSuccess,NSError *error))handle;
```

3.74 Sleep protocol (get a day to today)

```
/*!
 *  @func   Get all sleep data from a certain day to today
 *  @param  fromDayIndex  The number of days from today, (0: means today, 1: means
yesterday)
 *  @param  items      Returned sleep data (key: days from today, value: corresponding
sleep data)
 *  @param  fail        failed callback
 */
+ (void)getSleepDetailDataFromDay:(NSInteger)fromDayIndex sleepDatas:(nullable void (^)
(NSDictionary <NSString*,NSArray<QCSleepModel*>*>*_Nonnull))items fail:(nullable void
(^)(void))fail;
```