Development Guide

Engineering configuration

- Import the framework package

Import CRPSmartBand.framework into the project embedded binary file.

Development considerations

- Configuration permission

Configuring permissions in Info.plist:

Add App streamings using CoreBluetooth under Required background modes (Bluetooth Permissions)
Add Allow Arbitrary Loads under App Transport Security Settings and set YES (network access)
Add Bluetooth Permissions for Privacy - Bluetooth Always Usage Description

1 Data model

1.1 CRPDiscovery

Modifier	Name	Explain
String	LocalName	Device name
Int	RSSI	Rssi
String	Mac	Mac address

1.2 StepModel

Modifier	Name	Explain
Int	steps	steps
Int	distance	distance
Int	calory	calory
Int	time	time (When the value is greater than -1, it is the exercise statistics time, and the value -1 means that the device does not support exercise time statistics)

1.3 SleepModel

Modifier	Name	Explain
Int	deep	Deep sleep
Int	light	Light sleep
[Dictionary<String,String>]	detail	Sleep data details

Sleep data details

Modifier	Name	Explain
String	type	state 0: Awake 1:Light 2: Deep
String	total	Length of sleep (Minute))
String	start	start time
String	end	end time

1.4 ProfileModel

Modifier	Name	Explain
Int	height	height
Int	weight	weight
Int	age	age
GenderOption	gender	gender

1.5 AlarmModel

Modifier	Name	Explain
Int	id	Id(only 0，1，2)
Int	enable	Enable 0: off 1: on
AlarmType	type	repeat type
Int	hour	hour
Int	minute	minute
Int	year	year
Int	month	month
Int	day	day
[AlarmWeekDay]	weekday	E.g：[.sun, .thu, .mon]

1.6 HeartModel

Modifier	Name	Explain
Double	starttime	start Time
Double	endTime	end Time
[Int]	detail	A number per minute

1.7 SportModel

Modifier	Name	Explain
String	date	date
Int	starttime	start time
Int	endtime	end Time
Int	vaildTime	vaild Time
SportType	type	type of sport
Int	Step	stpes
Int	Distance	distance
Int	kcal	calory

1.8 weather

Modifier	Name	Explain
Int	type	Type of Weather
Int	temp	temperature
Int	pm25	PM2.5
String	festival	Festiveal
String	city	City

Tyep of Weather

CLOUDY	FOGGY	OVERCAST	RAINY	SNOWY	SUNNY	SANDSTORM	HAZE
0	1	2	3	4	5	6	7

1.9 forecastWeather

Modifier	Name	Explain
Int	type	Weather type
Int	MaxTemp	Maximum temperature
Int	MinTemp	lowest temperature

1.10 newVersionInfo

Modifier	Name	Explain
upgradeType	Type	Upgrade type
Stirng	version	Version information number of the new version
String	log	New version information
String	logEn	New version of English information
Int	mcu	When the chip type mcu = 4, it is HTX.

1.11 ScreenContent

Modifier	Name	Explain
ContentPosition	position	Position of the screen information 0: Up 1: Down
Content	upperContent	Above time
Content	underContent	Below the time
UIColor	contentColor	font color
String	MD	MD5 value of the image

1.12 watchFaceInfo

Modifier	Name	Explain
Int	id	watchFace number
Int	model	The serial number of the type to which the watchFace belongs
String	fileUrl	watchFace download path
String	imageUrl	Download path for the preview of the watchFace

1.13 watchFaceSupportModel

Modifier	Name	Explain
[Int]	supportModel	Support type data
Int	currentId	The serial number of the download watchFace market of the current band (0 or nil is the watchFace of the watchFace market that has not been downloaded in the current band)

1.14 Physiological

Modifier	Name	Explain
[reminderModel]	reminderModels	Reminder mode
Int	cycleTime	Number of days in the physiological cycle
Int	menstruationTime	Days of menstruation
Int	lastTimeMonth	Last menstrual month
Int	lastTimrDay	Last menstrual date
Int	remindTimeHour	hour
Int	remindTimeMintue	minute

1.15 SitRemindModel

Modifier	Name	Explain
Int	period	Sedentary reminder cycle (minutes)
Int	steps	Maximum number of steps
Int	startHour	Start time (24-hour clock)
Int	endHour	End time (24-hour clock)

1.16 eventRemind

Modifier	Name	Explain
Bool	isRemind	switch
Int	startHour	start Hour
Int	startMinute	start Minute
Int	remindCount	remind Count
Int	remindTimeInterval	Reminder interval (minutes)

1.17 drinkWaterRemind

Modifier	Name	Explain
Bool	isRemind	switch
Int	startHour	start Hour
Int	startMinute	start Minute
Int	remindCount	remind Count
Int	remindTimeInterval	Reminder interval (minutes)
Int	waterIntake	waterIntake

1.18 hrRemind

Modifier	Name	Explain
Bool	isRemind	switch
Int	max	heartrate

1.19 newVersionTpInfo

Modifier	Name	Explain
Stirng	tp_path	Upgrade file address
String	tp_md5	Upgrade file MD5
String	tp_offset	File upgrade address

1.20 contactProfileModel

Modifier	Name	Explain
Int	contactMax	Maximum number of contacts allowed to be synchronized
Int	contactAvatarWidth	The width of the contact avatar
Int	contactAvatarHeight	The height of the contact avatar

1.21 CRPContact

Modifier	Name	Explain
Int	contactID	Index
String	fullName	Name
UIImage	image	Avatar
String	selectPhoneNumber	phoneNumber

1.22 CRPHRVDataModel

Modifier	Name	Explain
Double	time	measure time
Int	sportLevel	Exercise intensity level
[Int]	detail	RRI data

1.23 CRPMedicineReminderModel

Modifier	Name	Explain
Int	medincineID	index
Int	pastDay	The number of days that the medicine has been taken
String	medincineName	Medicine name
Int	cycleTime	Medication cycle
[CRPMedicineTimeModel]	reminderTimes	Reminder time and number of medications

1.24 CRPHeartRecordModel

Modifier	Name	Explain
Int	remindTimeHour	hour
Int	remindTimeMinute	minute
Int	pillCount	Number of medicines taken

1.25 CRPStressRecordModel

Modifier	Name	Explain
Int	value	Measurement results
Int	time	timestamp

1.26 CRPExerciseGoalsModel

Modifier	Name	Explain
Int	steps	steps
Int	distance	diatace(m)
Int	kcal	kcal
Int	exerciseTime	ExerciseTime(minutes)

1.27 CRPExerciseGoalStateModel

Modifier	Name	Explain
Int	state	switch status
[Int]	weekday	Array of weekday (refer to CRPWeekday)

1.28 CRPNewSportModel

Modifier	Name	Explain
Int	id	index
Int	starttime	start time
Int	endtime	end time
Int	vaildTime	vaild Time
SportType	type	type of sport
Int	Step	stpes
Int	Distance	distance
Int	kcal	calory
[Int]	heartRate	heartRate

1.29 CRPSportRecord

Modifier	Name	Explain
Int	id	index
Int	startTime	start time
Int	type	type of sport

1.30 CRPEcardProfileModel

Modifier	Name	Explain
Int	eCardMax	Maximum count of ECards
Int	eCardURLLimit	Maximum length of ECard URL
[Int]	ids	sorted array of current ids

1.31 CRPECard

Modifier	Name	Explain
Int	id	index
String	title	title
String	url	url

1.32 CRPScreenMarketVersionModel

Modifier	Name	Explain
Int	apiVersion	api interface version
Int	funcVersion	Function version

1.33 CRPScreenTagModel

Modifier	Name	Explain
Int	tag_name	tag name
Int	id	Tag serial number
[CRPScreenModel?]	faces	dial

1.34 CRPScreenModel

Modifier	Name	Explain
String?	uploader	Designer name
String?	name	watch face name
Int	id	dial serial number
Int	download	Downloads
String?	file	File download link
Int	size	dial size
String?	preview	Preview link
String?	remark_lang	Language
String?	remark_cn	Chinese description
String?	remark_en	English description

1.35 CRPScreenModelInfo

Modifier	Name	Explain
String?	uploader	Designer name
String?	name	watch face name
Int	id	dial serial number
Int	download	Downloads
String?	file	File download link
Int	size	dial size
String?	preview	Preview link
String?	remark_lang	Language
String?	remark_cn	Chinese description
String?	remark_en	English description
[CRPScreenModel]	face_list	Similar watch faces list

1.36 CRPNewWatchFaceSupportModel

Modifier	Name	Explain
[Int]	supportModel	Supported type of data
Int	currentId	The serial number of the downloaded watch face market of the current watch (0 or nil means that the current watch has not downloaded the watch face of the watch face market)
Int	udtMaxSize	The size of the watch face market currently supported by the watch

1.37 CRPNewSportingModel

Modifier	Name	Explain
Int	startTime	start time
CRPSportState	type	sport state
Int	Step	Number of steps
Int	sportTime	sport time
Int	Distance	distance
Int	kcal	calories
Int	heartRate	real-time heart rate data

1.38 CRPStockSelectionModel

Modifier	Name	Explain
Int	id	serial number
Double	regularMarketOpen	opening price
Double	regularMarketDayHigh	Highest price
Double	regularMarketDayLow	lowest price
Int	regularMarketVolume	trading volume
Double	peRatio	Price-to-earnings ratio
Int	marketCap	market capitalization
Double	fiftyTwoWeekHigh	52-week high price
Double	fiftyTwoWeekLow	52-week low
Int	averageDailyVolume3Month	average trading volume
Double	regularMarketPrice	closing price
String	currency	currency
String	shortName	Abbreviation
String	symbol	keyword
String	exchange	exchange abbreviation
Double	regularMarketPreviousClose	Yesterday's closing price
Bool	isOpen	Whether to open

1.39 CRPCalendarModel

Modifier	Name	Explain
Int	id	Serial number
String	title	Title
Date	startTime	Start time
Date	endTime	End time

1.40 CRPCalendarConfigModel

Modifier	Name	Explain
Int	limit	Number of calendar synchronization supported
[CRPCalendarRecordModel]	records	Calendar records currently synchronized

1.41 CRPCalendarRemindModel

Modifier	Name	Explain
Bool	isRemind	Whether to remind
Int	remindTime	How many minutes in advance to remind

2 Agent method

2.1 Returns the current connection status of the band

```
didState(_ state: CRPState)
```

2.2 Return current Bluetooth status

```
didBluetoothState(_ state: CRPBluetoothState)
```

2.3 Receive real-time sports step data

```
receiveSteps(_ model: StepModel)
```

2.4 Receive heart rate measurement (single heart rate measurement)

```
receiveHeartRate(_ heartRate: Int)
```

2.5 Receive dynamic heart rate data

```
receiveHeartRateAll(_ model: HeartModel)
```

2.6 Receiving blood pressure measurement results

```
receiveBloodPressure(_ heartRate: Int, _ sbp: Int, _ dbp: Int)
```

2.7 Receive real-time heart rate data

Generally used only for display instead of storage, Some Bands support the return of real-time RRI data

```
receiveRealTimeHeartRate(_ heartRate: Int, _ rri: Int)
```

2.8 Receive blood oxygen measurement results

```
receiveSpO2(_ o2: Int)
```

2.9 Receive firmware upgrade progress and status

```
receiveUpgrede(_ state: UpgradeState, _ progress: Int)
```

2.10 Received a photo request

```
recevieTakePhoto()
```

2.11 Receive weather request (optional)

The band can save real-time weather for 2 hours, and the weather information will be cleared after 2 hours. When the band has no weather information today, when the band switches to the weather interface, it will request weather information via recevieWeather().

```
recevieWeather()
```

2.12 Received a request to find a phone (optional)

```
recevieFindPhone()
```

2.13 Receive real-time temperature data (optional)

After the wristband receives the real-time temperature value, please keep the last value received. When the measurement end status is received, the last value is the measurement result

```
receiveRealTimeTemperature(_ temp: Double)
```

2.14 Receive temperature measurement status (optional)

Used to judge the state of a single body temperature measurement 1：Measuring； 0：End of measurement

```
receiveTemperature(_ state: Int)
```

2.15 Receive a one-click call request (optional)

```
receiveCalling()
```

2.16 Receive HRV real-time data (optional)

```
receviceHRVRealTime(_ model: CRPHRVDataModel)
```

2.17 Receive current caller number (optional)

```
receivePhoneNumber(number: String)
```

2.18 Receive medication reminder data (optional)

```
receiveMedicineInfo(_ max: Int,_ model: CRPMedicineReminderModel?)
```

2.19 Receive sport mode status (optional)

When err is 1, it means that the startup failed, and the state returns to the end state when charging or low power; the watch is already exercising and returns to the type of exercise.

```
receiveSportState(_ state: SportType, _ err: Int)
```

2.20 Receive stress measurement results (optional)

```
receiveStress(_ stress: Int)
```

2.21 Receive the new version of the exercise record list (optional)

Callback data when the workout ends or when the getSportRecordList method is called

```
receiveSportList(_ list: [CRPSportRecord])
```

2.22 Receive GPS data record (optional)

When the getGPSDataRecordList method is called, the data is called back, and what is obtained is the start time list of the record, and the detailed data of the corresponding record can be obtained through the getGPSRecordData method

```
receiveGPSDataRecordList(time: [Int])
```

2.23 Receive GPS real-time fixed-point data (optional)

```
receiveGPSRealTime(location: CLLocationCoordinate2D)
```

2.24 Receive GPS auxiliary point request (optional)

When a GPS auxiliary point request is received, the syncGPSAuxiliary method needs to be called to send it

```
receiveGPSAuxiliaryRequest()
```

2.25 Receive GPS EPO data request (optional)

After receiving the GPS EPO data request, you need to call the startSyncEPO method to transmit the corresponding EPO data

```
receiveEPORequest(type: Int)
```

2.26 Receive GPS EPO synchronization progress and status (optional)

```
receiveEPOSync(_ state: CRPUpgradeState, _ progress: Int)
```

2.27 After receiving the connection confirmation status, use connectConfirm to initiate a connection confirmation, and return to the pairing status after the watch is selected (optional)

```
receiveConnectConfirm(_ state: CRPPairState)
```

2.28 Receive hrv measurements (optional)

```
receiveHRV(_ hrv: Int)
```

2.29 Receive motion data during exercise (optional)

```
receiveRealTimeSportData(data: CRPNewSportingModel)
```

2.30 Exit the camera interface after receiving the watch (optional)

Some watches support exiting camera interface notifications

```
receiveExitCameraView()
```

2.31 Receive stock request to update data (optional)

```
receiveStockRequestUpdateInfo()
```

2.32 Receive GPT request status or return data (optional)

```
receiveGPTState(type: CRPGPTType, state: CRPGPTRequestState, result: NSData)
```

2.33 Receive GPT request preview (optional)

```
receiveRequestGPTPreviewImage()
```

2.34 Receive GPT request background image (optional)

```
receiveRequestGPTImage()
```

3 Band scanning and connection

3.1 initialization

CRPSmartBand is the entry point of the SDK, Setting the delegate will initialize the CRPSmartBand and create Bluetooth (Tip: Creating Bluetooth is a time-consuming operation and needs to be set in advance)

```
CRPSmartBand.sharedInstance.delegate = self
```

3.2 Band scan

Normal scanning can be started with permission enabled and Bluetooth enabled. During the scanning process, the wristband is found. Through the scan() method, the band found in the scan through the progressHandler callback, the completionHandler adjusts all the bands scanned in the scan, and the scanning duration can be set. Because the Bluetooth scanning operation is time-consuming operation, it is recommended. Scan time is 10 seconds

```
scan(_ duration: TimeInterval = 10, progressHandler: scanProgressHandler?, completionHandler: scanCompletionHandler?);
```

- Get the system Bluetooth connected devices, get the list of the original CBPeripheral, you can use CBPeripheral to initialize CRPDiscovery and then connect.

```
getConnectedBand(_ handler: connectedPeripherialHandler)
```

3.3 Cancel scan

Cancel scan

```
interrupScan()
```

3.4 Band connection

By scanning [CRPDiscovery], you can select the appropriate band connection according to CRPDiscovery's localName and Mac address.

```
connet(_ discovery: CRPDiscovery)
```

3.5 Unbind

- Unlock the band

```
remove(_ handler: @escaping removeHandler)
```

- reconnect

```
reConnect()
```

- Dual-mode bluetooth release watch connection notification
  This method needs to be initiated before unbinding the watch to solve the problem that the Bluetooth pairing pop-up window may display slowly when connecting later

```
removeBond()
```

4 Usage

4.0 synchronised time

Keep the band and mobile phone time consistent

```
setTime()
```

4.1 Firmware upgrade

- Query firmware version

Query the current firmware version of the band

```
getSoftver(_ handler: @escaping stringHandler)
```

- Query new firmware

Get the new version of firmware upgradeable by passing in the current version and mac address

```
checkLatest(_ version: String, _ mac: String, handler: @escaping versionHandler)
```

- Query the DFU status of the wristband. When the DFU is not 0, the wristband is in the upgrade state.

```
checkDFUState(_ handler: @escaping stringHandler)
```

- Nordic firmware upgrade (HTX and Nordic firmware upgrade methods are different, you need to use the corresponding upgrade method according to the muc queryed, when the mcu is 4 or 8, use the HTX upgrade method to upgrade)

After receiving the new firmware callback, Nordic can call this interface to upgrade the opponent ring.

```
startUpgrede(info: newVersionInfo?)
```

- Nordic stops firmware upgrade

```
stopUpgrade()
```

- The Mac address when you get the OTA mode in HTX (the firmware upgrade of HTX must first obtain the Mac address when upgrading the firmware)

```
getOTAMac(_ handler: @escaping stringHandler)
```

- HTX firmware upgrade
  After receiving the new version firmware callback, use this interface to upgrade the opponent ring when mcu is 4 or 8.

```
startHTXOTA(info: newVersionInfo,upgradeMac: String, isUser: Bool, _ isPatch:Bool = true)
```

- HTX stops firmware upgrade

```
stopHTXOTA()
```

- RealTek firmware upgrade
  After receiving the callback of the new version of the firmware, use this interface to upgrade the opponent ring when the mcu is 7

```
startRealTekUpgrade(info: newVersionInfo?)
```

- RealTek TP upgrade
  After receiving the callback of the new TP version, use this interface to upgrade the opponent ring

```
startTpUpgrade(tpInfo: newVersionTpInfo?)
```

- Goodix Firmware Upgrade
  When mcu is 10, the Band is Goodix, use the checkLatest method to obtain the latest version, download the file of fileUrl, and then pass the file address into this method

```
startGoodixUpgradeFromFile(zipPath: String)
```

4.2 Query the power of the band

Query the current battery power

```
getBattery(_ handler: @escaping intHandler)
```

4.3 User Info

- Set user information

```
setProfile(_ profile: ProfileModel)
```

- Set step length
  In the band firmware 1.6.6 and above, you can set the step length to the band to calculate the activity data more accurately.

```
setStepLength(length: Int)
```

4.4 Weather

- Today's weather

  Set the weather today to the band.

```
setWeather(_ weather: weather)
```

- Weather in the next 7 days

  Set the weather for the next 7 days to the band

```
setForecastWeather(_ weathers:[forecastWeather])
```

- Set the current city name
  Some bracelets support setting the current city name

```
sendCityName(_ cityName: String)
```

- New version weather settings (support setting sunrise and sunset time)

```
setNewVerWeather(weather: CRPNewWeather)
```

4.5 Steps

- Today's steps

```
getSteps(_ handler:@escaping stepHandler)
```

- Get step count statistics
  Some hand-hands support the statistics of the last two days and half-hours.

  Get today's step statistics

```
get24HourSteps(_ handler: @escaping intsHandler)
```

Get yesterday's step statistics

```
get24HourSteps(_ handler: @escaping intsHandler)
```

- Get calorie statistics
  Some wristbands support half-hour classified statistics of calories in the last two days (data unit: small calorie)

Get today's calorie count

```
get24HourCals(_ handler: @escaping intsHandler)
```

Get yesterday's calorie count

```
getAgo24HourCals(_ handler: @escaping intsHandler)
```

- Get distance statistics
  Some wristbands support half-hour classification and statistics of distance in the last two days (data unit: cm)

Get today's distance stats

```
get24HourDistances(_ handler: @escaping intsHandler)
```

Get yesterday's distance statistics

```
getAgo24HourDistances(_ handler: @escaping intsHandler)
```

- Some watches support the acquisition of multi-day historical activity steps (day:0-6; 0 is today, 1 is yesterday, and so on)

```
getHistoryStepData(day: Int, handler: @escaping historyStpeHander)
```

4.6 Sleep

Get today's sleep data

```
getSleepData(_ handler:@escaping sleepHandler)
```

- Some watches support obtaining historical sleep data for multiple days (day: 0-6; 0 is today, 1 is yesterday, and so on)

```
getHistorySleepData(day: Int, handler: @escaping historySleepHandler)
```

4.7 Setting the time system

The band supports 12-hour and 24-hour systems.

- Set time system

```
setTimeFormat(_ type:Int)
```

- Query time system

  Query the time system that the band is using.

```
getTimeformat(_ handler: @escaping intHandler)
```

type: 0: 12-timeformat 1: 24-timeformat

4.8 Quick view

- Set the quick view

  Turns the quick view on or off.

```
setQuickView(_ isOn: Bool)
```

- Get the quick view

```
getQuickView(_ handler: @escaping intHandler)
```

0: close 1： open

- Set the effective time for quick view

```
setQuickViewTime(_ periodTime: periodTimeModel)
```

- Get the effective time for quick view

```
getQuickViewTime(_ handler: @escaping periodTimeHandler)
```

4.9 Goal steps

- Set goal steps

  Push the user's target step number to the band. When the number of active steps reaches the target step, the band will have a target to reach the reminder.

```
setGoal(_ goal: Int)
```

- Get goal steps

  Query the number of target steps that the band is set

```
getGoal(_ handler: @escaping intHandler)
```

4.10 Watchface

- Switch watchface

  The band supports three different watchface and can be switched freely.

```
setDial(_ type:Int)
```

type:

First watchFace	Second watchFace	Third watchFace
0	1	2

- Query the band watchFace

  Query the watchFace number that the band is using.

```
getDial(_ handler: @escaping intHandler)
```

- Replace the custom watchFace background image

The watchFace of the 1.3-inch color screen supports the replacement of the background picture. The picture size is 240 * 240 px. The upgrade process will call back the progress and status through the receiveUpgradeScreen.
imageSize and compressionType need to be obtained through the getScreenContent method

```
startChangeScreen(_ image: UIImage, _ imageSize :ScreenImageSize, _ isCricle: Bool = false, _ compressionType: Int)
```

- Modify the custom watchFace layout

```
setupScreenContent(content: ScreenContent)
```

The MD5 value of content needs to be set to the MD5 value of the background image. When the MD5 value is "", the background image restores the default background.

- Get the watchface layout

```
getScreenContent(_ handler: @escaping screenContentHandler)
```

- Get support watchface type

  Some bands have a new watchFace, query the supported watchFace type and the ID of the current table watchFace in the watchFace market

```
getWatchFaceSupportModel(_ handler: @escaping watchFaceSupportHandler)
```

- Get the watchface store

  Obtain a list of watchFaces that can be replaced according to the support type of the watchFace
  currentPage: the current page (starting from 1); perPage: how many dials are returned per page (recommended to be set to 18)

```
getWatchFaceInfo(_ supportModel: [Int], currentPage: Int, perPage: Int ,_ handler: @escaping watchFaceHandler)
```

- Sending a watchface file

  Send the watch face file to the band, during which the band will restart, and the upgrade process will call back the progress and status through receiveUpgradeScreen.

  watchFaceID: The watch on the JieLi platform needs to send the ID of the watch face file to complete the watch face transmission.

```
startChangeWatchFace(_ Info: watchFaceInfo, _ watchFaceID: Int = 0)
```

- Get the dial market space size
  Some watches have restrictions on the size of the dial. To obtain the dial, you need to pass in the size limit to obtain the dial size.

```
getUDTMaxSize(_ handler: @escaping intHandler)
```

- JieLi watch face market functional version
  The watch dials on the JieLi platform need to distinguish the supported dial market dials according to the functional version.

```
getJLScreenMarketVersionInfo(_ handler: @escaping screenMarketVersionInfoHandler)
```

- Get the new version watch face market tag list
  supportModel: the watch face template supported by the firmware (obtained by getWatchFaceSupportModel or getNewWatchFaceSupportModel method);
  ver: watch firmware version number;
  currentPage: current page (starting from 1);
  perPage: How many watch faces are returned per page (recommended setting is 18);
  memorySize: Use getUDTMaxSize or getNewWatchFaceSupportModel to obtain it. If there is no callback, just pass 0;
  platform: can be obtained during Bluetooth scanning;
  model: JieLi platform needs to use getJLScreenMarketVersionInfo to obtain it. For other platforms, apiVersion and funcVersion can be set to 0;

```
getNewWatchFaceTagList(supportModel: [Int], ver: String, currentPage: Int, perPage: Int, memorySize: Int, platform:
CRPPlatform, model: CRPScreenMarketVersionModel, _ handler: @escaping watchFaceTagListHandler)
```

- Get the list of watch faces under the single label of the new version of the watch face market
  tag_id: tag id;
  ver: watch firmware version number;
  supportModel: the watch face template supported by the firmware (obtained by getWatchFaceSupportModel or getNewWatchFaceSupportModel method);
  currentPage: current page (starting from 1);
  perPage: How many watch faces are returned per page (recommended setting is 18);
  memorySize: Use getUDTMaxSize or getNewWatchFaceSupportModel to obtain it. If there is no callback, just pass 0;
  platform: can be obtained during Bluetooth scanning;
  model: JieLi platform needs to use getJLScreenMarketVersionInfo to obtain it. For other platforms, apiVersion and funcVersion can be set to 0;

```
getNewWatchFaceList(tag_id: Int, ver: String, supportModel: [Int], currentPage: Int, perPage: Int, memorySize: Int,
platform: CRPPlatform, model: CRPScreenMarketVersionModel, _ handler: @escaping watchFaceListHandler)
```

- Get the details of the new version of the watch face market
  screenId: the watch face id to be queried;
  ver: watch firmware version number;
  supportModel: watch face template supported by the firmware (obtained by getWatchFaceSupportModel method);
  memorySize: Use getUDTMaxSize or getNewWatchFaceSupportModel to obtain it. If there is no callback, just pass 0;
  platform: can be obtained during Bluetooth scanning;
  model: JieLi platform needs to use getJLScreenMarketVersionInfo to obtain it. For other platforms, apiVersion and funcVersion can be set to 0;

```
getNewWatchFaceInfo(screenId: Int, ver: String, supportModel: [Int], memorySize: Int, platform: CRPPlatform, model:
CRPScreenMarketVersionModel, _ handler: @escaping watchFaceInfoHandler)
```

4.11 Alarm

- Set Alarm

  The band can support three alarm clocks, and the alarm information can be set separately according to the alarm number. A single alarm supports setting the date, and other repeating types of alarms are not supported.

```
setAlarm(_ alarm: AlarmModel)
```

- Get all alarm

  Query the alarm clock information saved by the band

```
getAlarms(_ handler: @escaping alarmHandler)
```

- new version alarm clock

Get all alarm information

```
getNewVerAlarm(_ handler: @escaping alarmHandler)
```

set an alarm

```
setNewVerAlarm(_alarm: AlarmModel)
```

Add an alarm

```
setNewVerAddAlarm(_ alarm: AlarmModel)
```

delete alarm

```
setNewVerDeleteAlarm(_ id: Int)
```

delete all alarms

```
setNewVerDeleteAllAlarm()
```

4.12 Language

- Set the band Language

The band supports multiple languages

```
setLanguage(_ lang: Int)
```

lang:

English	简体中文	日本語	한국어	Deutsch	Français	Español	العربية	Русский	繁體中文	Українська	Italiano	Português	Nederlands	Polskie	Svenska	Suomalainen	Dansk	Norsk	Magyar	Česky	български	Românesc	Slovenského jazyk
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Note: Please confirm the language supported by the bracelet, and the language that the bracelet does not support will be invalid.

- Get the current band language

```
getLanguage(_ handler: @escaping intHandler)
```

4.13 Message push

- Set other message push status

Turn on or off other message.

```
setNotification(_ switchs:[NotificationType])
```

- Get other message

  Query other message enable status

```
getNotifications(_ handler: @escaping notificationsHandler)
```

- New version of message push settings

Query the message push type supported by the watch, and know the message push type supported by the watch according to the returned list against NotificationType

```
getNotificationSupportType(_ handler: @escaping notificationsHandler)
```

Query the status of the new version message push type

```
getNotificationState(_ handler: @escaping notificationsHandler)
```

Set the new version message push type status

```
setNotificationState(_ states: [Int])
```

4.14 Sedentary reminder

- Set sedentary reminder

  Turn the sedentary reminder on or off.

```
setRemindersToMove(_ isOn: Bool)
```

- Get sedentary reminder

  Query the sedentary reminder status

```
getRemindersToMove(_ handler: @escaping intHandler)
```

- Set sedentary reminder time

Set the sedentary reminder effective time period

```
setSitRemind(_ sitRemind: SitRemindModel )
```

- Get sedentary reminder time

Get sedentary reminder time

```
getSitRemindInfo(_ handler: @escaping sitRemindHandler)
```

4.15 Find the band

Look for the band and the band will vibrate for a few seconds after receiving this command.

```
setFindDevice()
```

4.16 Heart rate

All heart rate related data will be called back via receiveRealTimeHeartRate(_ heartRate: Int)

- Measuring dynamic heart rate

  Dynamic heart rate measurement is started

```
setStartHeart()
```

- Stop dynamic heart rate measurement

The dynamic heart rate measurement is stopped and the measurement result is called back byreceiveHeartRateAll(_ model: HeartModel).

```
setStopHeart()
```

- Query last dynamic heart rate measurement

The dynamic heart rate is measured in an unconnected state and the band can save the last measurement. The result is called back byreceiveHeartRateAll

```
getHeartData()
```

- Set timing to measure heart rate

  The wristband supports 24-hour timed measurement of heart rate, starting from 0:0, you can set the measurement interval (0 is off, positive is on, and set to measurement interval: positive * 5 minutes)

```
set24HourHeartRate(_ interval:Int)
```

- Get 24-hour timing measurement status

```
get24HourHeartRateInterval(_ handler: @escaping intHandler)
```

- Query today to measure heart rate data regularly

```
get24HourHeartRate(_ handler: @escaping intsHandler)
```

- Query yesterday to measure heart rate data regularly

```
getAgo24HourHeartRate(_ handler: @escaping intsHandler)
```

- Get exercise data
  Some hand ring supports the heart rate measurement of multiple sports modes. The measurement results include heart rate and other motion related data such as calorie. This interface is used to obtain calories, etc. data. Save the last three sports data.

```
getSportData(_ handler: @escaping sportHandler)
```

- Turn on static heart rate single measurement

```
setStartSingleHR()
```

- End static heart rate single measurement
  End a single measurement. The measurement time is too short, resulting in no measurement data. The measurement result is called back by receiveHeartRate(_ heartRate: Int).

```
setStopSingleHR()
```

- Get a single heart rate measurement history record (supported by some bracelets)

```
getHeartRecordData()
```

- Some watches support obtaining multiple days of historical scheduled heart rate measurement data (day: 0-6; 0 is today, 1 is yesterday, and so on)

```
getHistoryHrData(day: Int, handler: @escaping historyHrHandler)
```

4.17 Blood pressure

- Start measuring blood pressure

```
setStartBlood()
```

- Stop measuring blood pressure

Stop measuring blood pressure, too short a measurement time will result in no measurement results. The measurement results are called back by receiveBloodPressure(_ heartRate: Int, _ sbp: Int, _ dbp: Int).

```
setStopBlood()
```

Some Band support all-day blood pressure function

- Set all-day blood pressure status

```
setFullDayBPStatus(open: Bool)
```

- Get all-day blood pressure status

```
getFullDayBPStatus(_ handler: @escaping intHandler)
```

- Get all-day blood pressure Data

```
getFullDayBPData(_ handler: @escaping intsHandler)
```

- Get a single blood pressure measurement history record (supported by some bracelets)

```
getBPRecordData()
```

4.18 Blood oxygen

- Start measuring blood oxygen

```
setStartSpO2()
```

- Stop measuring blood oxygen

  Stop measuring blood oxygen, too short a measurement time will result in no measurement results. The result is called back by receiveSpO2(_ o2: Int).

```
setStopSpO2()
```

Some Band support blood oxygen function throughout the day

- Set all-day blood oxygen status

```
setFullDayO2Status(open: Bool)
```

- Get all-day blood oxygen status

```
getFullDayO2Status(_ handler: @escaping intHandler)
```

- Get all-day blood oxygen Data

```
getFullDayO2Data(_ handler: @escaping intsHandler)
```

- Get the history of a single blood oxygen measurement (supported by some bracelets)

```
getO2RecordData()
```

4.19 Calibrate GSensor

- Calibrate GSensor

  If the wristband is not sensitive or the step is not accurate, the GSensor can be calibrated to repair. During the calibration process, the bracelet is placed horizontally on the table.

```
sendCalibration()
```

4.20 Take a photo

- Switch to the camera interface

```
switchCameraView()
```

- Set photo monitor

  Long press the band photo interface, you can trigger the camera's camera command, callback through recevieTakePhoto ()

4.21 Shut down

The band is turned off.

```
shutDown()
```

4.22 Do not disturb

- Set the do not disturb time

  The band supports the Do Not Disturb period, and no message push is displayed during the Do Not Disturb period.

```
setDisturbTime(_ periodTime: periodTimeModel)
```

- Query do not disturb time

  Check the do not disturb time set by the band.

```
getDisturbTime(_ handler: @escaping periodTimeHandler)
```

4.23 Historical steps and sleep data

The band can save the first two days of step data and sleep data (allDataHandler contains stepModels and sleepModels, the first of the array is yesterday data, the second is the previous day data)

- Get historical data

```
getAllData(_ handler:@escaping allDataHandler)
```

4.24 Physiological cycle

Some bands support setting physiological cycle reminders

- Set the physiological cycle reminder

```
setPhysiological(physiological: Physiological)
```

- Query physiological cycle reminder

```
getPhysiological(_ handler: @escaping physiologicalHandler)
```

4.25 ECG measurement

Some Band support ECG measurement

- Start ECG measurement

```
startECGMeasure()
```

- Stop ECG measurement

```
stopECGMeasure()
```

- Check if it is a new ECG measurement method
  In the new measurement method, the band can save the last unsent measurement result; the old version does not.

```
isNewECGMeasurementVersion()
```

- Query last ECG Data
  Query the ECG data saved in the band

```
getLastMeasureECGData()
```

- Set heart rate during cardiac measurement
  Using the data obtained from the measured amount, the instantaneous heart rate is calculated by the ECG algorithm library and sent to the band.

```
sendECGHeartRate(heartRate: Int)
```

4.26 Body temperature

Some bracelets support body temperature measurement

- Start a single temperature measurement
  The receiveRealTimeTemperature callback method returns the measurement value, and the receiveTemperature callback method returns the measurement status

```
sendSingleTemperatureStart()
```

- End a single temperature measurement

```
sendSingleTemperatureEnd()
```

- Get automatic temperature measurement switch
  1：open；0：close

```
getAutoTemperatureState(_ handler: @escaping intHandler)
```

- Set automatic temperature measurement switch

```
sendAutoTemperature(_ isOpen: Bool)
```

- Obtain automatic temperature measurement data
  One data every half hour

```
getAutoTemperatureData(_ todayHandler: @escaping doublesHandler, _ yesterdayHandler: @escaping doublesHandler)
```

- Get temperature unit
  1: Fahrenheit; 0: Celsius

```
getTemperatureUnit(_ handler: @escaping intHandler)
```

- Set temperature unit
  1: Fahrenheit; 0: Celsius

```
setTemperatureUnit(_ unit:Int)
```

Some bracelets support the all-day new version of the all-day body temperature function (measured once a minute)

- Set all-day body temperature status

```
setFullDayTemperatureStatus(open: Bool)
```

- Get all-day body temperature status

```
getFullDayTemperatureStatus(_ handler: @escaping intHandler)
```

- Get all-day body temperature Data

```
getFullDayTemperatureData(_ handler: @escaping intsHandler)
```

4.27 Bright screen time

Some bracelets support bright screen time selection

- Set bright screen time

```
sendAutoLockTime(_ time: Int)
```

- Get the bright screen time

```
getAutoLockTime(_ handler: @escaping intHandler)
```

4.28 Heart rate reminder

Some bracelets support heart rate reminder

- Set the heart rate reminder value

```
sendHeartRateRemind(_ remind: hrRemind)
```

- Get the heart rate reminder value

```
getHeartRateRemind(_ handler: @escaping hrRemindHandler)
```

4.29 Drink water reminder

Some bracelets support drinking reminders

- Set drinking reminder

```
sendDrinkWaterRemind(_ remind: drinkWaterRemind)
```

- Get drinking reminder

```
getDrinkWaterRemind(_ handler: @escaping drinkWaterHandler)
```

4.30 Hand washing reminder

Some bracelets support hand washing reminders

- Set hand washing reminder

```
sendHandWashRemind(_ remind: eventRemind)
```

- Get hand washing reminderG

```
getHandWashRemind(_ handler: @escaping reminderHandler)
```

4.31 Address book

Some bracelets support the function of synchronizing contacts

- Get contact configuration items

```
getContactProfile(_ handler: @escaping contactProfileHandler)
```

- Set up contacts

```
setContact(profile: contactProfileModel, contacts: [CRPContact]
```

- Clear contacts

```
cleanAllContact()
```

- Delete specified contact

```
deleteContact(contactID: Int)
```

- Get the current number of contacts

```
getContactCount(_ handler: @escaping intHandler)
```

- Get the current contact support version (0: the contact number does not support any symbols, 1: the contact number supports the transmission of "+", "*", "#" three symbols. If there is no reply, the contact number does not support any symbol)

```
getContactSupportVer()
```

4.32 Reset

Some bracelets support factory reset

- Reset

```
reset()
```

4.33 HRV

Some bracelets support HRV

- Set measurement interval

```
setHRV(_ interval:Int)
```

- Get measurement interval

```
getHRVInterval(_ handler: @escaping intHandler)
```

- Get the amount of fatigue data for a certain day in the historical data (dayIndex: 0~6, 7 days of data can be obtained, 0: today, 1: yesterday, and so on)

```
getHRVCount(_ dayIndex: Int, _ handler: @escaping intHandler)
```

- Obtain the fatigue data of a specified day in the historical data

```
getHRVData(_ dayIndex: Int, _ index: Int, _ handler: @escaping hrvDataHandler)
```

- Active measurement to obtain RRI data

```
startHRVMeasure()
```

- Stop measuring RRI data

```
stopHRVMeasure()
```

4.34 Power saving mode

Power saving mode for some Band

- Set power saving mode

```
setPowerSaveState(open: Bool)
```

- Get power saving mode status

```
getPowerSaveState(_ handler: @escaping intHandler)
```

4.35 Reminder to take medicine

Some bracelets support medication reminder

- Get the current medication reminder information, and return the data through the receiveMedicineInfo method

```
getMedicineInfo()
```

- Set a reminder to take medicine

```
setMedicine(medicine: CRPMedicineReminderModel)
```

- Delete the medication reminder of the specified id

```
deleteMedicine(id: Int)
```

- Delete all medication reminder settings

```
deleteAllMedicine()
```

4.36 Touch the screen to wake up

Some bracelets support setting the screen to wake up by tapping

- Set screen touch to wake

```
setTapToWake(open: Bool)
```

- Get the screen touch wake setting (0: off, 1: on)

```
readTapToWake(_ handler: @escaping intHandler)
```

4.37 Turn on sport mode

Part of the bracelet supports turning on the sport mode

- Set sport mode

```
setSportState(state: SportType)
```

Some bracelets support the new version of exercise data

- Get a list of exercise records
  Return data through the receiveSportList callback method

```
getSportRecordList()
```

- Get exercise details
  According to the list returned by receiveSportList, when it is judged that startTime>0, call the following method to obtain the detailed data of the corresponding id

```
getSportRecordData(id: Int, _ handler: @escaping sportDetailHandler)
```

4.38 Stress measurement

Some bracelets support stress measurement

- Query whether stress measurement is supported

```
getStressIsSupport(_ handler: @escaping intHandler)
```

- Start stress measurement, the measurement result is returned by the receiveStress(_ stress: Int) callback

```
setStartStress()
```

- stop stress measurement

```
setStopStress()
```

- Get stress measurement records

```
getStressRecord(_ handler: @escaping stressRecordDataHandler)
```

- Get all-day pressure measurement switch (0: off, 1: on)

```
getAutoStressInterval(_ handler: @escaping intHandler)
```

- Set the all-day pressure measurement switch (0: off, 1: on)

```
setAutoStress(_ interval:Int)
```

- Get today's all-day stress data

```
getTodayAutoStressData(_ handler: @escaping intsHandler)
```

- Get yesterday's all-day stress data

```
getYesterDayAutoStressData(_ handler: @escaping intsHandler)
```

4.39 Exercise goals

Some watch support exercise goals

- Get your daily exercise goals

```
getNormalExerciseGoal()
```

- Set your daily exercise goals

```
setNormalExerciseGoal(_ goals: CRPExerciseGoalsModel)
```

- Get exercise Day Goals

```
getExerciseDayGoal()
```

- Set exercise Day Goals

```
setExerciseDayGoal(_ goals: CRPExerciseGoalsModel)
```

- Get exercise Day Settings Status

```
getExerciseState()
```

- Set exercise Day Settings Status

```
setExerciseState(_ state: CRPExerciseGoalStateModel)
```

4.40 ECard

Some bracelets support ECards

- Get ECard configuration information

```
getECardProfile(handler: @escaping eCardProfileHandler)
```

- Set up ECard
  Set the ECard according to the obtained ECards configuration information

```
setECard(profile: CRPEcardProfileModel, eCard: CRPECard)
```

- Get ECard

```
getECardInfo(id: Int, _ handler: @escaping eCardInfoHandler)
```

- Delete ECard

```
deleteECard(id: Int)
```

- ECard sorting

```
setECardSequenceArray(ids:[Int])
```

4.41 GPS function

Some wristbands support GPS function

- Get the list of GPS data records, and return the record start time list through the receiveGPSDataRecordList(time: [Int]) method

```
getGPSDataRecordList()
```

- Get detailed data of GPS records
  Use the start time returned by receiveGPSDataRecordList(time: [Int]) to obtain the correspondence, and locate a point every two seconds

```
getGPSRecordData(time: Int, _ handler: @escaping gpsDataHandler)
```

- Start syncing EPO data

```
startSyncEPO(path: String, type: Int)
```

- Synchronized GPS assistance points
  When the wristband turns on GPS for exercise, it will request auxiliary positioning points from the App through receiveGPSAuxiliaryRequest()

```
syncGPSAuxiliary(local: CLLocationCoordinate2D)
```

4.42 Audio media switch

Part of the wristbands audio media switch (0: not connected to call Bluetooth, 1: A2DP connected, 2: A2DP disconnected, 0xFF: does not support A2DP control)

- Set audio media state

```
setA2DPControl(state: Int)
```

- Get audio media state

```
getTalkBluetoothConnectState(_ handler: @escaping intHandler)
```

- get audio bluetooth name

```
getTalkBluetoothName(_ handler: @escaping stringHandler)
```

4.43 Vibration intensity

Some wristbands support setting the vibration intensity (1: weak, 2: medium, 3: strong)

- Set vibration intensity

```
setVibrationStrength(type: Int)
```

- Get vibration intensity

```
getVibrationStrength(handler: @escaping intHandler)
```

4.44 Sleep detection time setting

Some wristbands support sleep detection time setting (restricted time period: start sleep time 20~3, end 6~12 o'clock)

- Set sleep detection time

```
setSleepTime(info: CRPSleepDetectionInfo)
```

- Get sleep detection time

```
getSleepTime(handler: @escaping sleepDetectionInfoHandler)
```

4.45 Stock

Some watches support stock functions

- Get stock support quantity

```
getStockSupportInfo(_ handler: @escaping intHandler)
```

- Set stocks to no information

```
setNullStockData()
```

- Set or update stock information

```
setStockData(data: CRPStockSelectionModel)
```

- Delete stocks with specified id

```
deleteStock(id: Int)
```

- Set display stock sorting

```
setStockSequence(ids: [Int])
```

4.46 GPT

Some watches support GPT function

- Get GPT support type

```
getGPTSupportType(_ handler: @escaping gptSupportTypeHandler)
```

- Set GPT status

```
setGPTState(type: CRPGPTType, state: CRPGPTReplyState)
```

- Set GPT issues by parsing the result audio data of receiveGPTState

```
setGPTQuestion(type: CRPGPTType,content:String)
```

- Reply with questions and answers

```
setGPTAnswer(content:String)
```

- Transfer AI dial image preview
  Can only be initiated after the receiveRequestGPTPreviewImage method callback

```
setGPTPreImage(image: UIImage)
```

- Transfer AI watch face pictures
  It can be initiated actively or after receiveRequestGPTImage callback.

```
startGPTScreen(_ image: UIImage, _ imageSize :ScreenImageSize, _ isCricle: Bool = false, _ compressionType: Int)
```

- Modify AI dial layout

```
setupGPTScreenContent(content: ScreenContent)
```

The MD5 value of content needs to be set to the MD5 value of the background image. When the MD5 value is "", the background image returns to the default background.

- Get AI dial layout

```
     getGPTScreenContent(_ handler: @escaping screenContentHandler)
```

4.47 Calendar

Some watches support calendar synchronization

- Get calendar configuration

```
getCalendarConfig(handler: @escaping calendarConfigHandler)
```

- Set calendar

```
setCalendar(calendar:CRPCalendarModel)
```

- Get calendar information
  The calendar function is mainly based on the App

```
getCalendar(id: Int, handler: @escaping calendarHandler)
```

- Delete the specified id calendar

```
deleteCalendar(id: Int)
```

- Delete all calendars

```
deleteAllCalendar()
```

- Set calendar reminder settings

```
setCalendarReminder(model: CRPCalendarRemindModel)
```

- Get calendar reminder settings

```
getCalendarReminder(handler: @escaping calendarRemindHandler)
```

4.48 Scan code pairing

Some watches support scan code pairing. When the scan_code_bind property of CRPDiscovery is true, it means that the watch supports the scan code binding function

- Set the pairing random code. The pairing status will be returned through receiveConnectConfirm

```
connectConfirmByRandomCode(randomCode:Int)
```

- App actively triggers the pairing result

```
setPairReply(allow: Bool)
```

5 Version update log

2.5

Add step number count interface

2.6

Add physiological cycle related interface

2.7

Remove non-essential third-party library dependencies (Alamofire)

2.8

Modify the packaged version to Xcode11-swift5.1

2.9

Add ECG measurement method

3.1

Add sedentary reminder to set effective time

3.7

Add temperature function

3.8

Step data adding time

3.9

Add the function of synchronizing contacts

3.10

Add HRV measurement function

3.11

Added a new version of all-day blood oxygen, blood pressure, body temperature measurement functions

3.12

Add medicine reminder function

3.13

Add to get a single measurement record and touch the screen to wake up the switch

3.14

Add open sport mode

3.15

Add pressure measurement
Add new version of exercise data
Add card function

3.16

Add GPS function
Compatible with RealTek8763 firmware upgrade
Add vibration intensity
Sleep detection time setting
New version weather setting, new version alarm clock setting and new version message push setting
Add all-day pressure measurement function
Adapt to JieLi platform
Added access to sports data during exercise

3.16.26
increase stock
Add GPT function

3.18.1

增加扫码配对相关方法
增加日历相关接口