Development guide

1 Environment configuration

1.1 Add AAR package

Copy my-ble-*.aar to the libs directory of the project.

1.2 Declare permissions

Configure permissions in AndroidManifest.xml.

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

 <!-- Android 12 Add new permissions-->
<uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
```

1.3 Code obfuscation

```
-keep class com.crrepa.ble.** { *; }
```

2 Usage

2.1 Initialization

CRPBleClient is the entrance to the SDK. The client needs to maintain an instance of CRPBleClient. It is recommended to initialize CRPBleClient in Application.onCreate().

```
CRPBleClient bleClient = CRPBleClient.create(context);
```

2.2 Scan

1. Scan

Normal scanning can only start when permissions are allowed and Bluetooth is turned on. Callback via CRPScanCallback.onScanning() when a ring is found during scanning. After the scan is completed, all rings found during the scan are called back through CRPScanCallback.onScanComplete(). Scan duration can be set,unit is milliseconds. Because the Bluetooth scanning operation is time-consuming, it is recommended that the scanning time be 10 seconds.

```
bleClient.scanDevice(scanCallback, 10000);
```

2. Cancel scan

Stopping the scan during the scan and canceling the scan will also trigger CRPScanCallback.onScanComplete().

```
bleClient.cancelScan();
```

2.3 Connect

1. Connect

Query the Mac Address of the ring through the CRPScanDevice of the scan callback. Establish a connection with the ring through CRPBleDevice.connect(), and call back the connection state through CRPBleConnectionStateListener.onConnectionStateChange(). It is recommended that when disconnecting and reconnecting, an appropriate delay can be added so that the system can recycle resources and ensure the connection success rate.

```
String address = CRPScanDevice.getDevice().getAddress();
CRPBleDevice bleDevice = bleClient.getBleDevice(address);
CRPBleConnection bleConnection = bleDevice.connect();
bleConnection.setConnectionStateListener(bleConnectionStateListener);
```

2. Reconnection

After receiving the disconnect callback, reconnection can be initiated. The reconnection method is as follows:

- Create a new GATT connection (**recommended**)

  Close the current GATT connection and re-establish a new GATT connection.

```
// Close GATT connection
CRPBleConnection.close();
// It is recommended to delay for three seconds before initiating the
connection.
String address = CRPScanDevice.getDevice().getAddress();
CRPBleDevice bleDevice = bleClient.getBleDevice(address);
CRPBleConnection bleConnection = bleDevice.connect();
bleConnection.setConnectionStateListener(bleConnectionStateListener);
```

- Reconnect via current GATT

```
CRPBleConnection.connect();
```

2.4 Sync time

Synchronize the phone time to the ring.

```
bleConnection.syncTime();
```

2.5 User info

Set user information

```
bleConnection.sendUserInfo(CRPUserInfo);
```

CRPUserInfo:

weight	height	gender	age	stepsLength
Weight (unit: kg)	Height (unit: cm)	Gender (male is 0; female is 1)	Age	Step length (unit: cm)

2.6 Firmware

1. Query firmware version

```
bleConnection.queryFrimwareVersion(CRPFirmwareVersionCallback);
```

Query firmware hash

```
bleConnection.queryFirmwareHash(CRPFirmwareHashCallback callback);
```

2. Query new firmware

New firmware information is passed through the CRPNewFirmwareVersionCallback.onNewVersion() callback.

```
// version is the current firmware version.
bleConnection.checkFirmwareVersion(CRPDeviceNewFirmwareVersionCallback, version,
CRPFirmwareCheckType);
```

CRPFirmwareCheckType:

NORMAL	BETA	FORCED
Upgrade normally. Used for regular upgrades.	BETA upgrade. Used for internal beta version upgrade.	Forced upgrade. Generally not used, used for major updates to force users to upgrade.

CRPFirmwareVersionInfo:

version	changeNotes	changeNotesEn	type	mcu
Current firmware version number	Change log	English update log	Upgrade method, same as CRPFirmwareCheckType	MCU type, used to distinguish upgrade methods

3. Start OTA (firmware upgrade)

The OTA progress is called back through CRPOtaListener.

```
bleConnection.startOta(CRPOtaListener);

// or
bleConnection.startOta(File, CRPOtaListener);
```

4. Abort OTA

After the OTA is successful or failed, this interface can be called to exit the firmware upgrade process.

```
bleConnection.abortOta();
```

5. Query OTA status

Query whether the ring is in the DFU data transmission state. In the new version of firmware, the ring will restart multiple times during the OTA process. When the ring is in the DFU state, avoid sending other instructions to the ring. The result is passed to the CRPOtaStateCallback.onDfuState() callback.

```
bleConnection.queryOtaState(CRPDeviceDfuStatusCallback);
```

CRPDeviceDfuStatusCallback

DEVICE_STATUS_NORMAL	DEVICE_STATUS_DFU
Normal status	DFU status

2.7 Battery

1. Query battery

When the ring's battery exceeds 100, it means the ring is charging. Through the CRPBatteryListener.onBattery() callback.

```
bleConnection.queryBattery();
```

2. Set up a listener

```
bleConnection.setBatteryListener(CRPBatteryListener);
```

3. Query real-time battery and voltage

Via CRPBatteryListener.onRealTimeBattery() callback.

```
bleConnection.queryRealTimeBattery();
```

2.8 Activity

1. Set up a listener

All activity-related data will be called back through CRPStepsChangeListener.

```
bleConnection.setStepChangeListener(CRPStepsChangeListener);
```

2. Query today's steps

The results are passed through the CRPStepsChangeListener.onCurrentSteps() callback.

```
bleConnection.queryCurrentSteps();
```

CRPStepsInfo:

steps	distance	calories	time
Steps	Distance (unit: meters)	Calories (unit: kcal)	Activity duration (unit: seconds)

3. Query historical steps

Query the number of active steps on a certain day. The result is passed through the CRPActionChangeListener.onHistoryStepChange() callback.

```
bleConnection.queryHistorySteps(CRPHistoryDay);
```

4. Query historical steps details

Query the details of the number of steps on a certain day. The number of steps throughout the day is calculated at half-hour intervals. The result is passed through the CRPStepsChangeListener.onHistoryStepsDetails() callback.

```
bleConnection.queryHistoryStepsDetails(CRPHistoryDay);
```

5. Set up a listener

```
bleConnection.setActionDetailsListener(CRPActionDetailsListener listener);
```

6. Query activity details

Through the CRPActionDetailsListener.onActionDetails() callback.

```
bleConnection.queryActionDetails(CRPHistoryDay historyDay);
```

2.9 Sleep

1. Set up a listener

All sleep-related data will be called back through CRPSleepChangeListener.

```
bleConnection.setSleepChangeListener(CRPSleepChangeListener);
```

2. Query sleep

Query the sleep data of a certain day, and the result is called back through CRPSleepChangeListener.onHistorySleepChange(). The ring sleep clearing time is 8 pm, and the ring records the sleep time period from 8 pm to 10 am the next day.

```
bleConnection.queryHistorySleep(CRPHistoryDay);
```

CRPSleepInfo：

totalTime	deepTime	lightTime	awakeTime	remTime	details	SLEEP_STATE_AWAKE	SLEEP_STATE_LIGHT	SLEEP_STATE_DEEP	SLEEP_STATE_REM
Total sleep time	Deep sleep time	Light sleep time	Waking hours	REM time	Sleep details	Waking state	Light sleep state	Deep sleep state	REM state

CRPSleepInfo.DetailBean:

startTime	endTime	totalTime	type
Start time	End Time	Total time	Sleep type

2.10 Heart rate

1. Set up a listener

All heart rate related data will be called back through CRPHeartRateChangeListener.

```
bleConnection.setHeartRateChangeListener(CRPHeartRateChangeListener);
```

2. Enable timing heart rate

Measurement starts at 0:00, and the measurement time interval can be set to multiples of 5 minutes.

```
// Measurement interval = interval * 5 minutes
bleConnection.enableTimingHeartRate(interval);
```

3. Disable timing heart rate

```
bleConnection.disableTimingHeartRate();
```

4. Query timing heart rate state

Query the state of scheduled heart rate measurement, and the result is called back through CRPHeartRateChangeListener.onTimingInterval().

```
bleConnection.queryTimingHeartRateState();
```

5. Query history timing heart rate

Query the heart rate data of the previous day, and the result is called back through CRPHeartRateChangeListener.onTimingHeartRate().

```
bleConnection.queryHistoryTimingHeartRate(CRPHistoryDay);
```

6. Start measure heart rate

Start measuring a single heart rate, and the result is called back through CRPHeartRateChangeListener.onHeartRate().

```
bleConnection.startMeasureHeartRate();
```

7. Stop measure heart rate

End single measurement. If the measurement time is too short, there will be no measurement data.

```
bleConnection.stopMeasureHeartRate();
```

8. Query history heart rate

Through CRPHeartRateChangeListener.onHistoryHeartRate() callback.

```
bleConnection.queryHistoryHeartRate();
```

2.11 Training

1. Set up a listener

```
bleConnection.setTrainingListener(CRPTrainingChangeListener);
```

2. Query support training type

Through CRPTrainingChangeListener.onSupportTrainingList() callback

```
bleConnection.querySupportTraining();
```

3. Start training

```
bleConnection.startTraining(type, CRPTrainingGoalsInfo);
```

4. Modify training state

```
bleConnection.sendTrainingState(CRPTrainingState state, CRPTrainingGoalsInfo
info);
```

5. Query training state

Through the CRPTrainingChangeListener.onTrainingState() callback.

```
bleConnection.queryTrainingState();
```

6. Query history training

History records through CRPTrainingChangeListener.onHistoryTrainingChange() callback.

```
bleConnection.queryHistoryTraining();
```

7. Query the details of the new version of training

History records through CRPTrainingChangeListener.onTrainingChange() callback.

```
// id historical record sequence number, starting from 0 and increasing
sequentially
bleConnection.queryTraining(id);
```

8. Achieve training goals

After reaching the goal during training, call back through CRPTrainingChangeListener.onGoalsReached(CRPGoalsType).

2.12 Blood oxygen

1. Set up a listening

All blood oxygen related data will be called back through CRPBloodOxygenChangeListener.

```
bleConnection.setBloodOxygenChangeListener(CRPBloodOxygenChangeListener);
```

2. Start measuring blood oxygen

```
bleConnection.startMeasureBloodOxygen();
```

3. Stop measuring blood oxygen

Stop measuring blood oxygen. If the measurement time is too short, there will be no measurement result. The result is passed through the CRPBloodOxygenChangeListener.onBloodOxygen() callback.

```
bleConnection.stopMeasureBloodOxygen();
```

4. Enable timing blood oxygen measurement

```
// interval represents the measurement interval, measurement interval = 5 * interval (minutes)
bleConnection.enableTimingBloodOxygen(interval);
```

5. Disable timing blood oxygen measurement

```
bleConnection.disableTimingBloodOxygen();
```

6. Query timing blood oxygen measurement state

Through CRPBloodOxygenChangeListener.onTimingInterval() callback.

```
bleConnection.queryTimingBloodOxygenState();
```

7. Query history timing blood oxygen measurement results

Via CRPBloodOxygenChangeListener.onTimingBloodOxygen() callback.

```
bleConnection.queryHistoryTimingBloodOxygen(CRPHistoryDay);
```

8. Query single blood oxygen history record

Via CRPBloodOxygenChangeListener.onHistoryBloodOxygen() callback.

```
bleConnection.queryHistoryBloodOxygen();
```

2.13 HRV (Heart rate variability)

1. Set up a listener

```
bleConnection.setHrvChangeListener(CRPHrvChangeListener);
```

2. Query whether to support new HRV

```
bleConnection.querySupportNewHrv();
```

3. Start measuring HRV

```
bleConnection.startMeasureHrv();
```

4. Stop measuring HRV

```
bleConnection.stopMeasureHrv();
```

5. Query history HRV records

```
bleConnection.queryHistoryHrv();
```

6. Enable timing measurement HRV

```
bleConnection.enableTimingHrv();
```

7. Disable timing measurement HRV

```
bleConnection.disableTimingHrv();
```

8. Query timing HRV state

Via CRPHrvChangeListener.onTimingInterval() callback.

```
bleConnection.queryTimingHrvState();
```

2.14 Activity goals

1. Set up a listener

```
bleConnection.setGoalsListener(CRPGoalsListener);
```

2. Set daily goals

Daily goals are set daily target values.

```
bleConnection.sendDailyGoals(CRPDailyGoalsInfo);
```

CRPDailyGoalsInfo:

steps	calories	trainingTime	distance
Steps	Calories (unit: kcal)	Training duration (unit: minutes)	Distance (unit: meters)

3. Query daily goals

Via CRPGoalsListener.onDailyGoals() callback.

```
bleConnection.queryDailyGoals(CRPDailyGoalsCallback);
```

4. Set training day goals

Training day is to mark a certain day as an training day. The target value can be set separately. The target value on the training day will overwrite the daily target value.

```
bleConnection.sendTrainingDayGoals(CRPDailyGoalsInfo);
```

5. Query training day goals

Through CRPGoalsListener.onTrainingDayGoals() callback.

```
bleConnection.queryTrainingDayGoals();
```

6. Goal achieved

Through the CRPGoalsListener.onGoalsAchieved(CRPGoalsType) callback.

2.15 Temperature

1. Set up a listener

```
bleConnection.setTempChangeListener(CRPTempChangeListener listener);
```

2. Enable timing temp

After the scheduled measurement is turned on, the ring automatically measures temperature every half hour.

```
bleConnection.enableTimingTemp();
```

3. Disable timing temp

```
bleConnection.disableTimingTemp();
```

4. Query timing temp state

Measurement state is passed through the CRPTempChangeListener.onTimingState() callback.

```
bleConnection.queryTimingTempState();
```

5. Query history temp

Through the CRPTempChangeListener.onHistoryTempChange() callback.

```
bleConnection.queryHistoryTemp(CRPHistoryDay);
```

2.16 Stress

1. Set up a listener

```
bleConnection.setStressChangeListener(CRPStressChangeListener);
```

2. Start measure stress

After the measurement is completed, call back through CRPStressChangeListener.onStressChange().

```
bleConnection.startMeasureStress();
```

3. Stop measure stress

```
bleConnection.stopMeasureStress();
```

4. Query history stress

Through the CRPStressChangeListener.onHistoryStressChange() callback.

```
bleConnection.queryHistoryStress();
```

2.17 Setting

1. Shut down

```
bleConnection.shutDown();
```

2. Reset

```
bleConnection.reset();
```

3. Set device information

```
bleConnection.sendSettingInfo(CRPSettingInfo);
```

CRPSettingInfo:

color	size	type
Device color	size	type

4. Query device information

Callback via CRPSettingCallback.onSetting().

```
bleConnection.querySettingInfo(CRPSettingCallback);
```

5. Set up wearing state change listener

When the wearing state of the ring changes, call back through CRPWearStateChangeListener.onWearStateChange().

```
bleConnection.setWearStateChangeListener(CRPWearStateChangeListener);
```

6. Query measurement state

Through the CRPMeasureStateCallback.onMeasuring() callback.

```
bleConnection.queryMeasureState(CRPMeasureStateCallback);
```

2.18 Activity reminder

1. Set up a listener

```
bleConnection.setActivityReminderListener(CRPActivityReminderListener);
```

2. Query activity reminder

Query the activity reminder status, and the result is called back through CRPActivityReminderListener.onActivityReminderDetails().

```
bleConnection.queryActivityReminder();
```

3. Set activity reminder

```
bleConnection.sendActivityReminder(CRPActivityReminderInfo);
```

CRPActivityReminderInfo:

period	steps	startHour	endHour	enable
Reminder period (unit: minutes)	Maximum number of steps	Start time (24-hour format)	End time (24-hour format)	Whether to turn on

4. Activity reminder

After reaching the reminder condition, call back through CRPActivityReminderListener.onActivityReminder().

2.19 Heart rate alert

1. Set heart rate alert

Set the heart rate alert state and alert heart rate value.

```
bleConnection.sendHeartRateAlert(enable, hr);
```

2. Query heart rate alert

Query the heart rate alert on state and heart rate alert value. The result is passed to the CRPHeartRateAlertCallback.onHeartRateAlert() callback.

```
bleConnection.queryHeartRateAlert(CRPHeartRateAlertCallback);
```

3 Change log