MEDIATEK

# Android BT Related Topic Explanation (SmartDevice APK)

2017.07.18

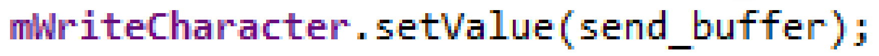
## Outline

- Android GATT MTU
- Android BT Permission
- GATT Write Type
- BLE Connection Parameter
- Workaround for Android BLE
- Android BT Log

# Android GATT MTU (1/3)

- Background
  - Sometimes APK needs to send large data to device by GATT, such as OTA.
  - GATT Client (Android APK) could request a larger MTU size to be able to send more data at once.
  - MTU Size = MTU Header (3 Bytes) + Data Size,  23 ~ 512 Bytes

- Android API
  - Default MTU: 23 Byte.  (i.e. Data size only 20 Bytes)



```
mWriteCharacter.setValue(send_buffer);
```

  - boolean requestMtu (int mtu)
  - A onMtuChanged(BluetoothGatt, int, int) callback will indicate whether this operation was successful.
  - Added in API level 21

## Android GATT MTU (2/3)

- Android Limitation
  - Android “requestMtu” API is added from API Level 21 (Android L, 5.0).
  - MTK platform SmartPhone + Android L could not support “requestMtu”.

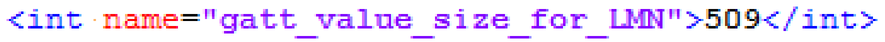
| | MTK SP | Non-MTK SP |
|---|---|---|
| Android KK (19, 4.4) | No | No |
| Android L (21/22, 5.0/5.1) | No | Yes |
| Android M (23, 6.0) or higher | Yes | Yes |

  - If device initiated to “requestMtu”, Android GATT Fwk will not callback “onMtuChanged” to APK. So APK cannot know whether this “requestMtu” was successful and how many final MTU is.

# Android GATT MTU (3/3)

- Wearable.jar
  - 1. Wearable.jar will "requestMtu" when BLE connected successfully in Android M+ SP and non-MTK L SP.

  - 2. Config "gatt send value size" (MTU-3, 20~509 Bytes)
  - Please refer to "App\src\main\res\xml\wearable_config.xml".

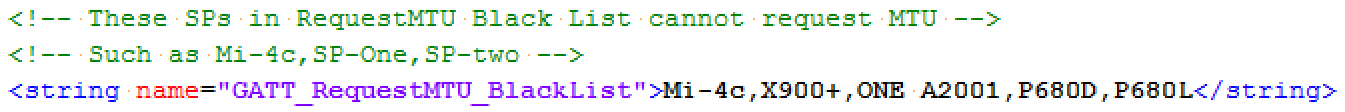


```
<int name="gatt_value_size_for_LMN">509</int>
```

  - 3. "requestMtu" Black List

A small number of phone could request MTU 512 successfully, but they cannot send large data normally.

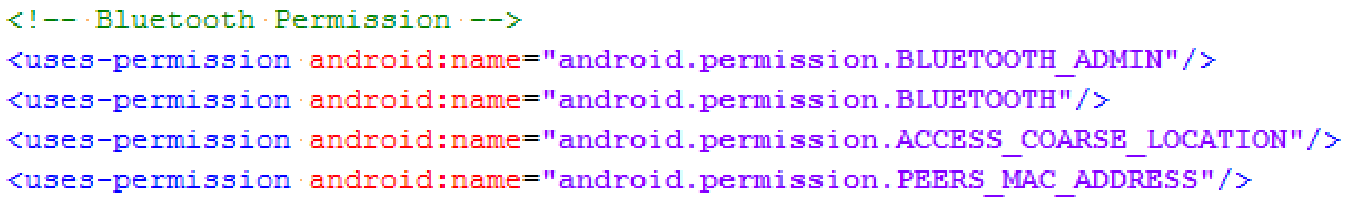
  - Wearable.jar will exclude  them, not "requestMtu" in these phones.



```
<!-- These SPs in RequestMTU Black List cannot request MTU -->
<!-- Such as Mi-4c,SP-One,SP-two -->
<string name="GATT_RequestMTU_BlackList">Mi-4c,X900+,ONE A2001,P680D,P680L</string>
```

## Android BT Permission (1/2)

- Permission in AndroidManifest
  - For BT(BLE) feature, please add below permissions in your Android Manifest xml.



```
<!-- Bluetooth Permission -->
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.PEERS_MAC_ADDRESS"/>
```

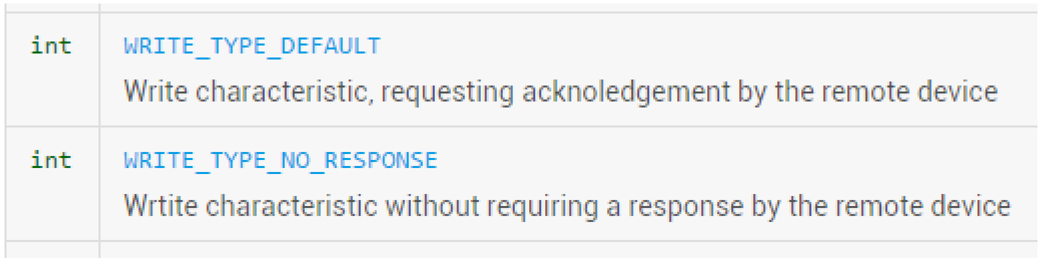
- Target SDK Level
  - If your app could not scan BLE device, but the Android BT setting could scan.
  - Please go to Android setting -> apps -> “your app” -> permission manage, then turn on “location” permission.
  - How to skip this problem?
  - A simple workaround: set TargetSDKVersion as 21

## Android BT Permission (2/2)

- Target SDK Level
  - “TargetSDKVersion=21” will disable “runtime permission” feature in Android M+ SP.
  - If your app must set TargetSDKVersion as 23+, or you don’t want to use this workaround.
  - You should refer to below webs to implement “request BT Permissions in Runtime” in Android M+ phone.
  - Google Android Web, Blog Web

# GATT Write Type (1/2)

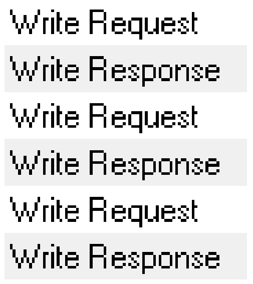
- Background
  - BluetoothGattCharacteristic could set two write type.
  - void setWriteType (int writeType)



| | |
|---|---|
| int | WRITE_TYPE_DEFAULT<br>Write characteristic, requesting acknoledgement by the remote device |
| int | WRITE_TYPE_NO_RESPONSE<br>Wrtite characteristic without requiring a response by the remote device |

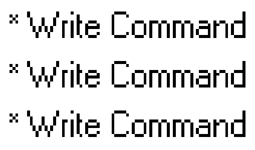
  - BT HCI Log

Write_Type_Default



```
Write Request
Write Response
Write Request
Write Response
Write Request
Write Response
```

Write_Type_No_Response



```
× Write Command
× Write Command
× Write Command
```

## GATT Write Type (2/2)

- Android IoT Issue
  - Theoretically, use “write_no_response” could accelerate up GATT sending rate.
  - But some smart phones will occur issue “no GATT Fwk callback” or “BLE disconnected suddenly”, when wearable.jar use “write_no_response” to send large data to device, such as OTA.
- Wearable.jar
  - Now, wearable.jar uses “Write_default_response” as write type from Linkit SDK 4.5.0.
  - Please use new wearable.jar with “Write_default_response”.
  - Old wearable.jar uses “Write_no_response” from Linkit SDK 4.0.0 to 4.3.0.

## BLE Connection Parameter (1/2)

- Background
  - Sometimes we needs to improve BLE rate in order to send large data quickly by GATT, such as OTA.
  - BLE connection parameter could be used to adjust BLE rate & power consumption for your device.

- Android API
  - boolean requestConnectionPriority (int connectionPriority)
  - CONNECTION_PRIORITY_BALANCED (default, connection paramters recommended by the Bluetooth SIG.)
  - CONNECTION_PRIORITY_HIGH (high rate, high power)
  - CONNECTION_PRIORITY_LOW_POWER (low rate, low power)
  - Added in API level 21

# BLE Connection Parameter (2/2)

- Android Limitation
  - "requestConnectionPriority" API is added from API Level 21 (Android L, 5.0).
  - The API will call native BT stack API, may have IOT(Interoperability Test) issue.

- Wearable.jar
  - "requestConnectionPriority" API is not recommended to use in your APP.
  - If you want to use, please refer to "// Request Connection Parameter" related code in "App\src\main\java\com\mtk\app\fota\UpdateFirmwareActivity.java".

- Device BT Code
  - The best way about speed up BLE rate is that adjust connection parameter in your device code.
  - Please refer to related "LinKit BT SDK" document.

## Workaround for Android BLE (1/3)

- BLE Scan Fail
  - Cause
    - The battery of your device may be too low.
    - There are many BT devices nearby.
    - Other unknown error.
  - Solution
    - Prompt User to do:
      - 1. Move to a place of less BT devices.
      - 2. Try in following turn: Kill your APK -> Reboot BT -> Reboot SP/Device, then restart.
      - 3. Enter Android Setting -> Apps -> select your APK -> Permission (manage) -> Enable "Location" permission.

# Workaround for Android BLE (2/3)

- Connect Fail
  - Cause
    - *BluetoothDevice getType* return unknown type.
    - Other unknown error on some SPs.
  - Solution
    - 1. Rescan when *BluetoothDevice getType* return unknown type
      - Need to check *BluetoothDevice getType* return value before *setRemoteDevice* or *connect*. If return *DEVICE_TYPE_UNKNOWN*, APK must rescan that device, then *setRemoteDevice* and *connect.*
    - 2. Retry to connect
      - If *onConnectChange* callback change *STATE_CONNECTING* state to *STATE_CONNECT_LOST/FAIL,* you could call *WearableManager connect* to retry.

## Workaround for Android BLE (3/3)

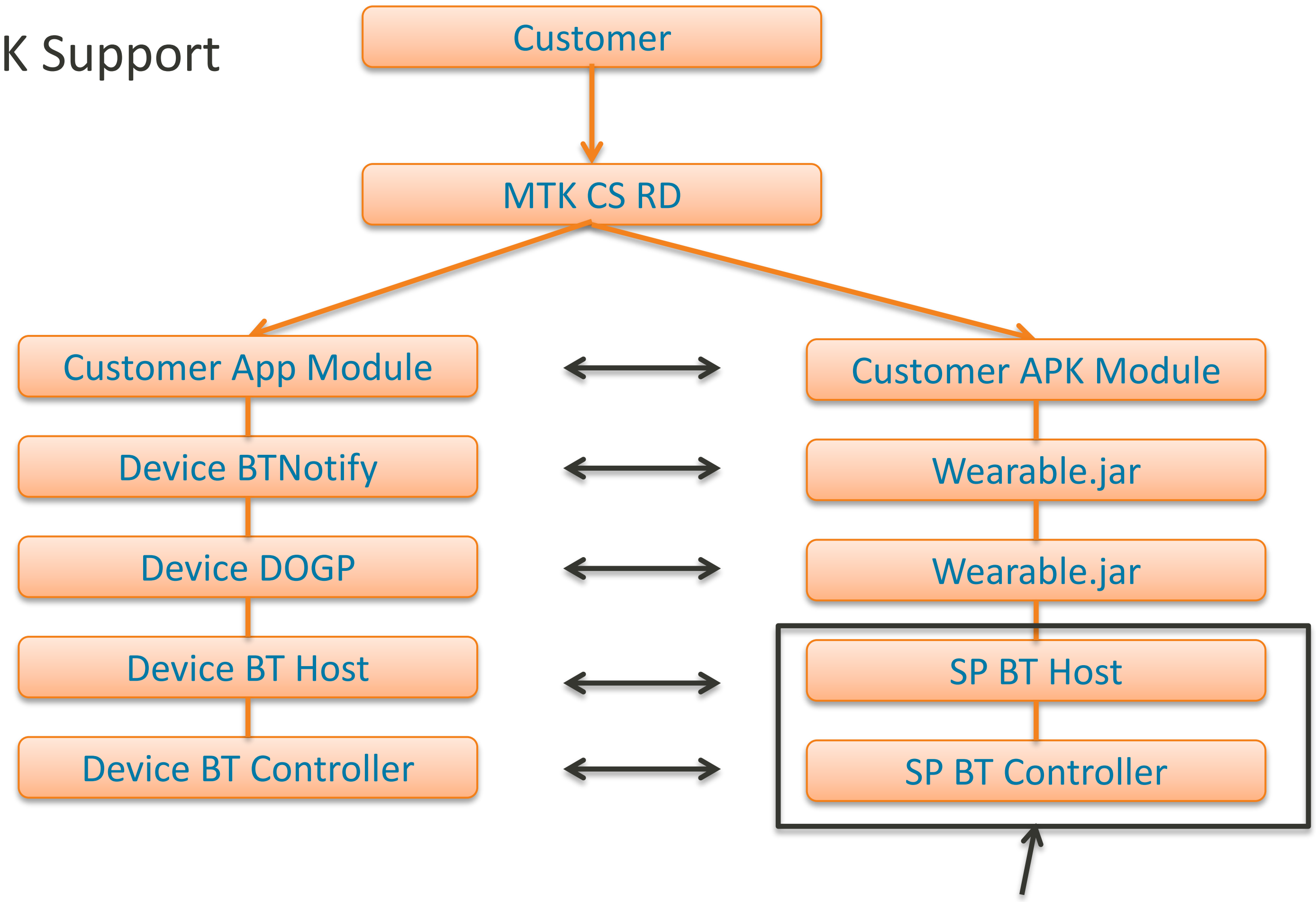
- Connect Fail
  - Solution
    - 3. Remove pairing if your device bonded with SP BT
      - Use reflect and call *BluetoothDevice* API *removeBond*, then retry to *connect.*
      - In order to prevent SP to show pairing confirm dialog, APK could call *BluetoothDevice createBond* and listen *ACTION_BOND_STATE_CHANGED.*
    - 4. Prompt User to do
      - "Try in following turn: Kill your APK -> Reboot BT -> Reboot SP/Device", then *connect*.

# Android BT Log (1/6)

- BT log
  - To analyze BT issue, Tester/Customer need to provide some log, include Android logcat log, Android BT HCI log, Device sys log, Device HCI log, BT air sniffer log.
  - For wearable.jar RD, must check Android logcat log & Android BT HCI log.
  - For Device App RD, they need device sys log.
  - For Device BTNotify/DOGP/Host RD, they need device sys log & device HCI log.
  - For Device Controller RD, they need analyze BT air sniffer log.
  - After Customer found a BT issue, you should capture these log and create a CR in MTK eService system or send to MTK customer support (CS) RD directly.

# Android BT Log (2/6)

- MTK Support



SP BT RDs are not in MTK IoT BU.

# Android BT Log (3/6)

- MTK Support
  - Customer only need to communicate with MTK CS RD.
  - MTK CS RD will communicate related MTK internal RD after they identify issue type and  analyze log.
  - Customer need to ensure code quality of your Device app & APK app modules.
  - Customer need to explain your develop requirement, reproduce step, reproduce ratio, which SP to MTK CS RD, or wrote in CR description.

- Device sys/HCI log
  - How to capture device sys and BT HCI log?  Please refer to related “Linkit SDK” document.

# Android BT Log (4/6)

- Android Logcat log
  - Capture Step
    - 1. Use “adb logcat -c” to clear previous useless log.
    - 2. Use “adb logcat –v threadtime > 1.log” to catch Android log with thread/time info to file “1.log”.
    - 3. Reproduce issue.
  - Problem
    - Some SP only could not catch Log in Log level D.
    - Solution:
    - Enter “engine mode” -> project menu -> Log setting -> turn on Log Level Debug and Verbose.
    - Such as Huawei SP

# Android BT Log (5/6)

- Android BT HCI Log
  - Capture Step
    - 1. Remove previous useless “btsnoop_hci.log” file in “sdcard/” or “internal storage” first folder by using your Android FileManager APP.
    - 2. Go to setting -> developer option UI.
    - 3. Reboot (disable -> enable) “Turn on Bluetooth HCI log” option.

    (Always Enable “Turn on Bluetooth debug log” option, if there is.)
    - 4. Reproduce issue.
    - 5. Use “adb pull” to export “btsnoop_hci.log”.

## Android BT Log (6/6)

- BT Air Sniffer Log
  - Need specific equipment (Ellisys).
  - If customer hasn’t the device, could only provide Android logcat, Android BT HCI log and device sys log to MTK.
  - After MTK RD analyzed log, if we cannot reproduce the issue and need BT air sniffer log, MTK Customer Support Team will support you to capture BT air sniffer log.

## Contact US

(If you have any questions, comments, or suggestions, please contact us by MTK ACS, or send mail to SmartDevice_App@mediatek.com)

MEDIATEK

everyday genius