Ben Berry
CrossComm, Inc.

BLE on Android

Bluetooth

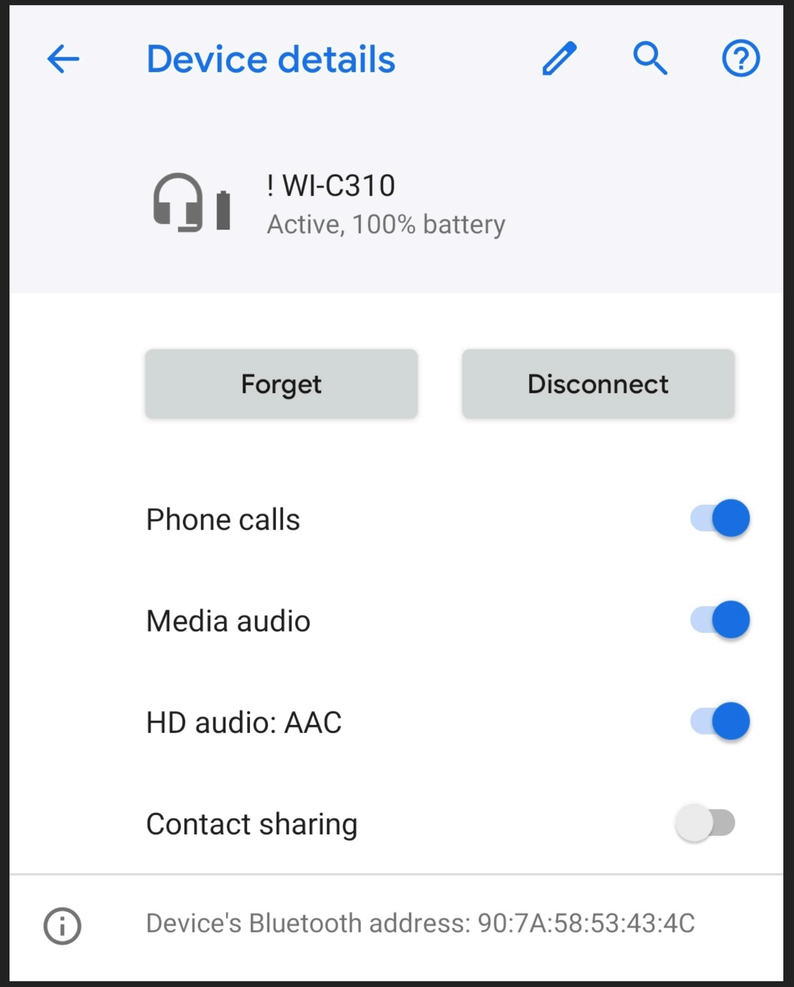
- Properties
  - Between WiFi and NFC
  - Always Transmitting
- Bluetooth 1.0 [1999]
  - Rough launch, conflicting implementations
- Bluetooth 1.2 [2003]
- Bluetooth 3.0 [2009]
  - “AMP”/“+HS” (?)
- Bluetooth 4.0. [2010]
  - Bluetooth Low Energy
  - Only transmit when you have something to say
  - 1-to-1, advertising or connected
- 4.1, 4.2, 5.0, 5.1, …
- Audio codecs completely separate
  - aptX, SBC, aptX HD, LDAC, AAC

Bluetooth Profiles

- GAP - Generic Access
- A2DP - Advanced Audio Distribution
- AVRCP - AV Remote Control
- HSP - Headset
- PBA - Phone Book Access
- HID - HID
- GATT - Generic Attribute

➡ Handshaking
➡ One-way Audio
➡ Play/Pause/Next
➡ Phone calls
➡ Showing contacts
➡ Keyboards & Mice
➡ BLE

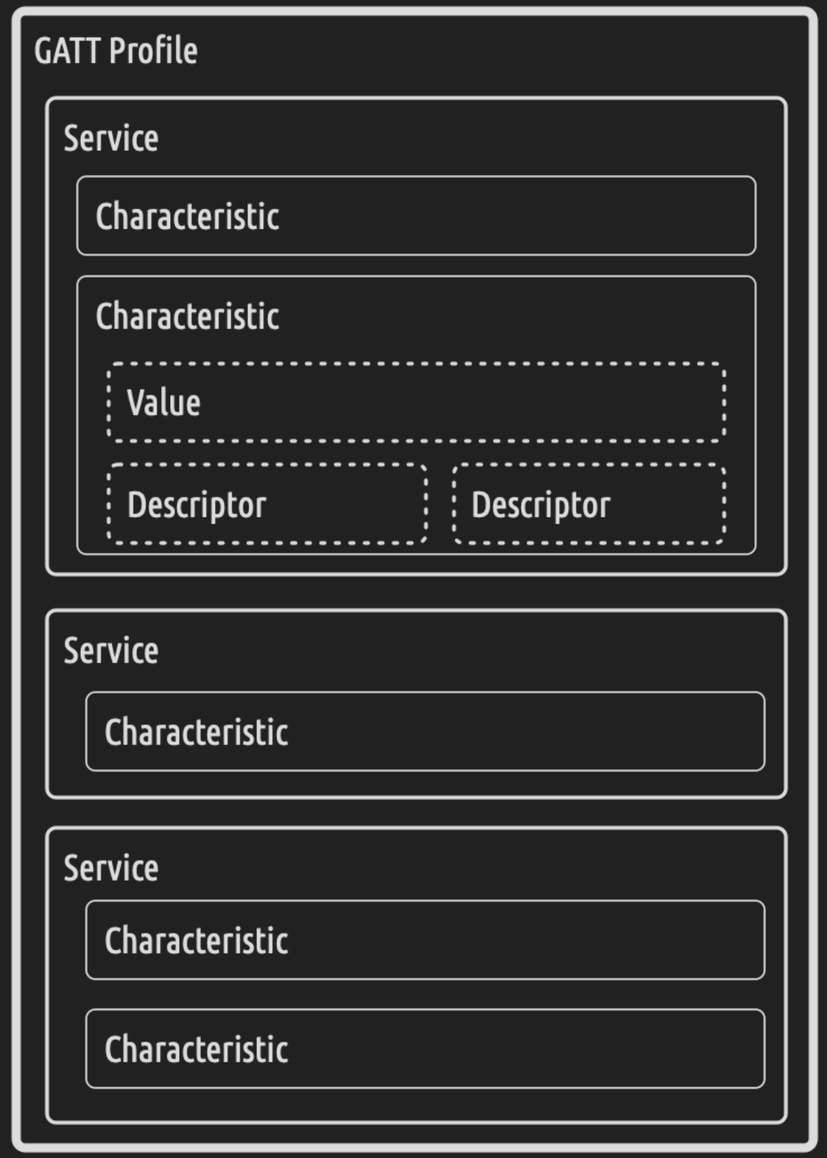
- GAP - Generic Access
- A2DP - Advanced Audio Distribution
- AVRCP - AV Remote Control
- HSP - Headset
- PBA - Phone Book Access
- HID - HID
- GATT - Generic Attribute



BLE Data

- Service
  - Characteristic
    - Value
    - Descriptor

- Known Services & Characteristics
- Max size: 512 bytes? 20 bytes? 40 bytes?
  - Max Transmission Unit (MTU)?
  - Exceed MTU?
- Read/Write
- Subscribe to notifications



Based on https://www.oreilly.com/library/view/getting-started-with/9781491900550/ch04.html

How to BLE

1. Scan for devices with filter (GAP)
2. Connect to Device (No Pairing)
3. Get GATT Client
4. Discover Services
5. Get Characteristic
   - Read Value
   - Read Descriptor(s)
   - Write Value
   - Subscribe to Change Notifications

Scanning

```
fun scanForDevices(context: Context, handler: ScanHandler) {
    val adapter = BluetoothAdapter.getDefaultAdapter()

    if (!adapter.isEnabled) {
        return handler(ScanEvent.ScanError, null, "Bluetooth interface disabled")
    }

    context.runWithPermissions(Manifest.permission.ACCESS_COARSE_LOCATION) {
        adapter.bluetoothLeScanner.startScan(callback)
    }
}
```

- ACCESS_COARSE_LOCATION?
- Add to the manifest:

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

Scanning with Filters

```
fun scanForDevices(context: Context, serviceUUID: UUID, handler: ScanHandler) {
    val adapter = BluetoothAdapter.getDefaultAdapter()

    if (!adapter.isEnabled) {
        return handler(ScanEvent.ScanError, null, "Bluetooth interface disabled")
    }

    val uuid = ParcelUuid(serviceUUID)
    val filter = ScanFilter.Builder().setServiceUuid(uuid).build()
    val filters = listOf(filter)

    val settings = ScanSettings
        .Builder()
        .setScanMode(ScanSettings.SCAN_MODE_LOW_LATENCY)
        .build()

    context.runWithPermissions(Manifest.permission.ACCESS_COARSE_LOCATION) {
        adapter.bluetoothLeScanner.startScan(filters, settings, callback)
    }
}
```

Handling Devices

```
private val callback = object : ScanCallback() {
    override fun onBatchScanResults(results: MutableList<ScanResult>?) {
        results?.forEach { result ->
            deviceFound(result.device)
        }
    }

    override fun onScanResult(callbackType: Int, result: ScanResult?) {
        result?.let { deviceFound(result.device) }
    }

    override fun onScanFailed(errorCode: Int) {
        handleError(errorCode)
    }
}

private fun deviceFound(device: BluetoothDevice) {
    device.connectGatt(context, true, gattCallback)
}
```

```
private val gattCallback = object : BluetoothGattCallback() {
    override fun onConnectionStateChange(gatt: BluetoothGatt?, status: Int, newState: Int) {
        if (newState == BluetoothGatt.STATE_CONNECTED) {
            gatt?.requestMtu(256)
            gatt?.discoverServices()
        }
    }

    override fun onServicesDiscovered(gatt: BluetoothGatt?, status: Int) {
        val characteristic = gatt?.getService(expandUuid(0x180F)) // Battery service
            ?.getCharacteristic(expandUuid(0x2A19)) // Battery life
        gatt?.readCharacteristic(characteristic)
        gatt?.setCharacteristicNotification(characteristic, true)
        characteristic?.value = byteArrayOf(50)
        gatt?.writeCharacteristic(characteristic)
    }

    override fun onCharacteristicRead(
        gatt: BluetoothGatt?, characteristic: BluetoothGattCharacteristic?, status: Int
    ) { /* ... */ }

    override fun onCharacteristicWrite(
        gatt: BluetoothGatt?, characteristic: BluetoothGattCharacteristic?, status: Int
    ) { /* ... */}

    override fun onCharacteristicChanged(
        gatt: BluetoothGatt?, characteristic: BluetoothGattCharacteristic?
    ) {
        characteristic?.let {
            val batteryLife = characteristic.value[0].toInt()
            Log.d(TAG, "Battery life is: $batteryLife")
        }
    }
}
```

Android Review

```
- BluetoothAdapter
      .getDefaultAdapter()
      .bluetoothLeScanner
      .startScan(scanCallback)

- onScanResult(result: ScanResult?) {
      result?.device.connectGatt(context, true, gattCallback)
  }

- gattCallback = object : BluetoothGattCallback() {
     override fun onConnectionStateChange(…) {
         gatt?.discoverServices()
     }
     override fun onServicesDiscovered(…) { }
     override fun onCharacteristicRead(…) { }
     override fun onCharacteristicChanged(…) { }
  }
```