



AsReader Finger-Type

Android SDK 開発マニュアル

ASR-023B

変更履歴

バージョン	変更内容	日付
1.0	新規作成	2020/7/10
1.1	1. getAddress を削除 2. receivedMacAddress を削除	2020/9/16
1.2	AsFingerConnectionType 値の修正	2024/07/01

目 次

概要	4
1 ライブラリーファイルのインポート及び使用	5
1.1 ライブラリーファイルをインポート	5
1.2 ライブラリーファイルを使う	8
1.3 プロジェクト権限を追加	13
2 メソッドの説明	15
2.1 AsFingerSDK	15
2.1.1 getInstance	15
2.1.2 setAsFingerEventListener	15
2.1.3 connect.....	17
2.1.4 setActivity	17
2.1.5 disconnect.....	17
2.1.6 getSdkVersion.....	18
2.1.7 getFirmwareVersion	18
2.1.8 getBattery	18
2.1.9 startScan	19
2.1.10 stopScan	19
2.1.11 sendData	19
2.1.12 startDiscovery	20
2.1.13 stopDiscovery	20
2.1.14 getPairedDevices.....	20
2.1.15 deviceType	21
2.1.16 connectedDevice	21
2.1.17 autoConnect.....	21
2.1.18 setAutoConnect	22
2.2 AsFingerEventListener	23
2.2.1 onStateChanged	23
2.2.2 receivedBarcodeData	23
2.2.3 receivedBattery	24
2.2.4 receivedFirmwareVersion	24
2.2.5 receivedData	24
2.2.6 receivedDevice	25
2.2.7 receivedFoundDeviceFinished	25

3 Enum	27
3.1 DeviceType	27
3.2 AsFingerConnectionType	27

概要

説明内容：

- 開発環境を構築、AsReader SDK ライブラリーで Android のアプリケーションを開発する方法を説明します。
- SDK ライブラリーの使い方を説明します。

開発ツール：

- Android Studio 3.5.2+
- Android SDK 10.0
- Android Gradle 5.4 - 2.1

システム要求：

- Android 9.0+

1 ライブラリーファイルのインポート及び使用

1.1 ライブラリーファイルをインポート

アプリケーションの目次にプロジェクトファイル「libs」を見つかり、右クリックして、「Reveal in Finder」を選択します。（図 1-1-1）

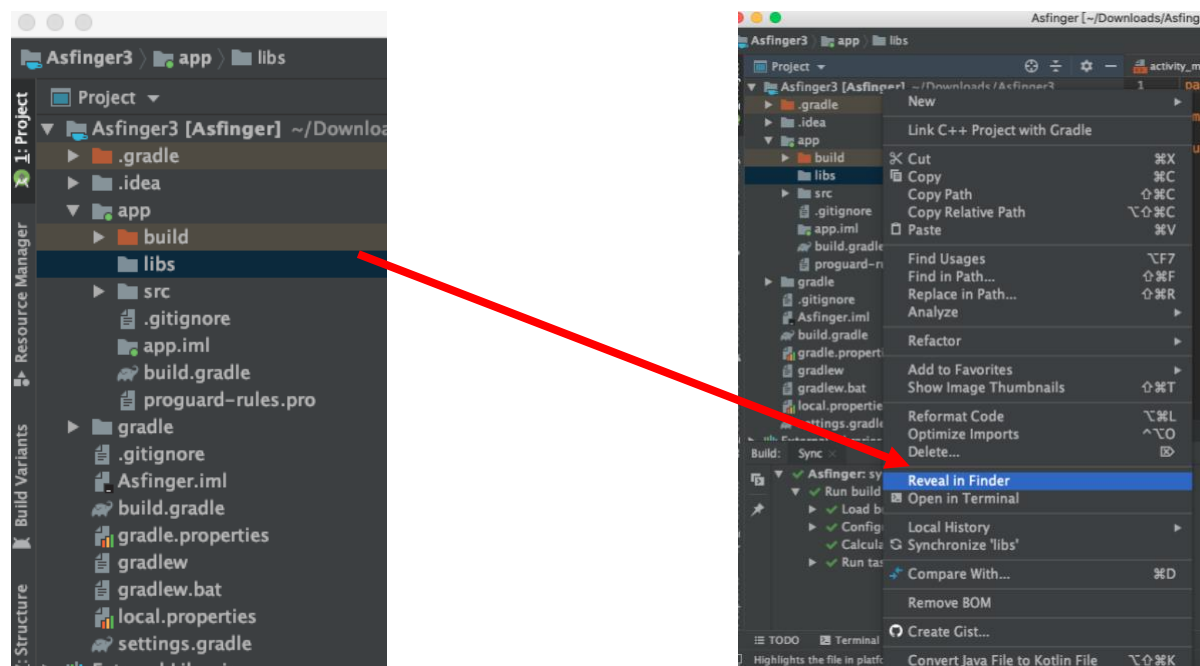


図 1-1-1

ポップアップしたウィンドウに「libs」を選択し、「asfingersdk.aar」を本フォルダにペーストします（図 1-1-2）。「asfingersdk.aar」は本プロジェクトの「libs」に表示されます（図 1-1-3）。

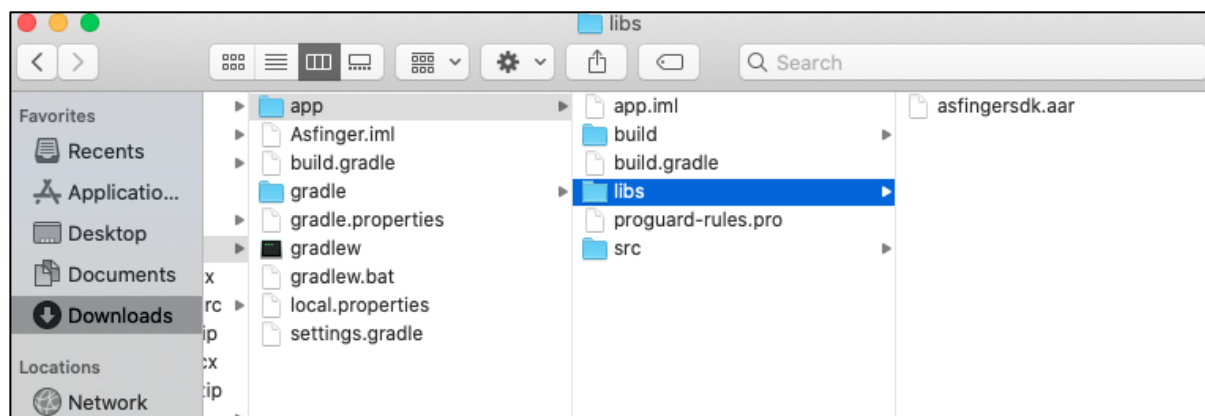


図 1-1-2

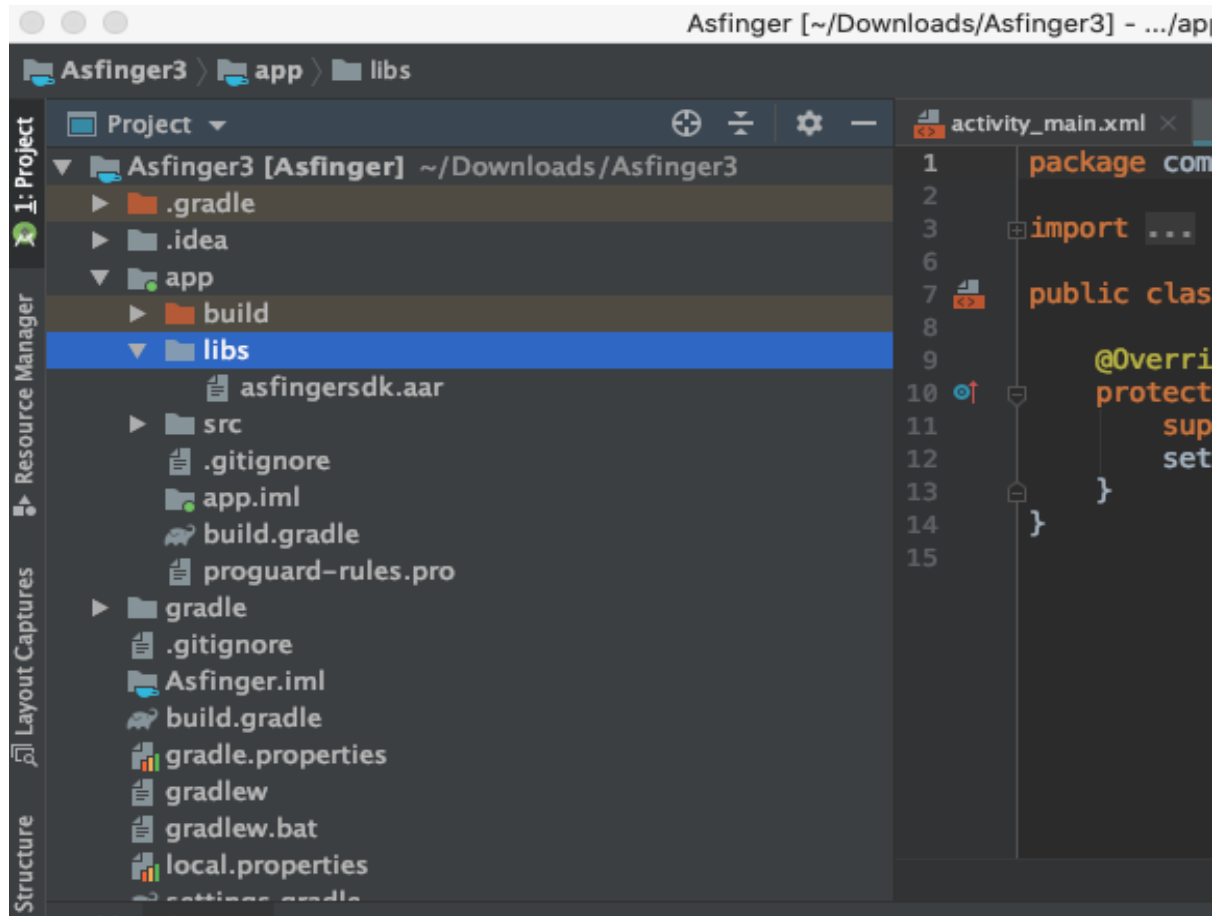


図 1-1-3

プロジェクトにある「build.gradle」をダブルクリックして（図 1-1-4）、図 1-1-5 のマーク 1、2 の通りに依存ライブラリーを追加し、マーク 3 の通りに「Sync Now」をクリックして、同期します。

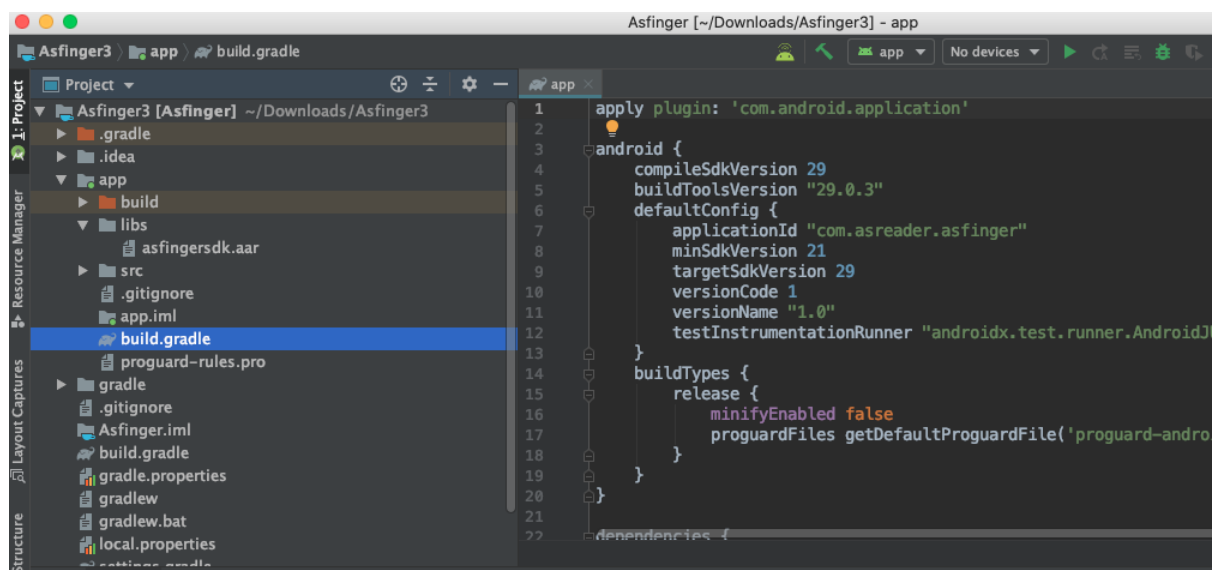


図 1-1-4

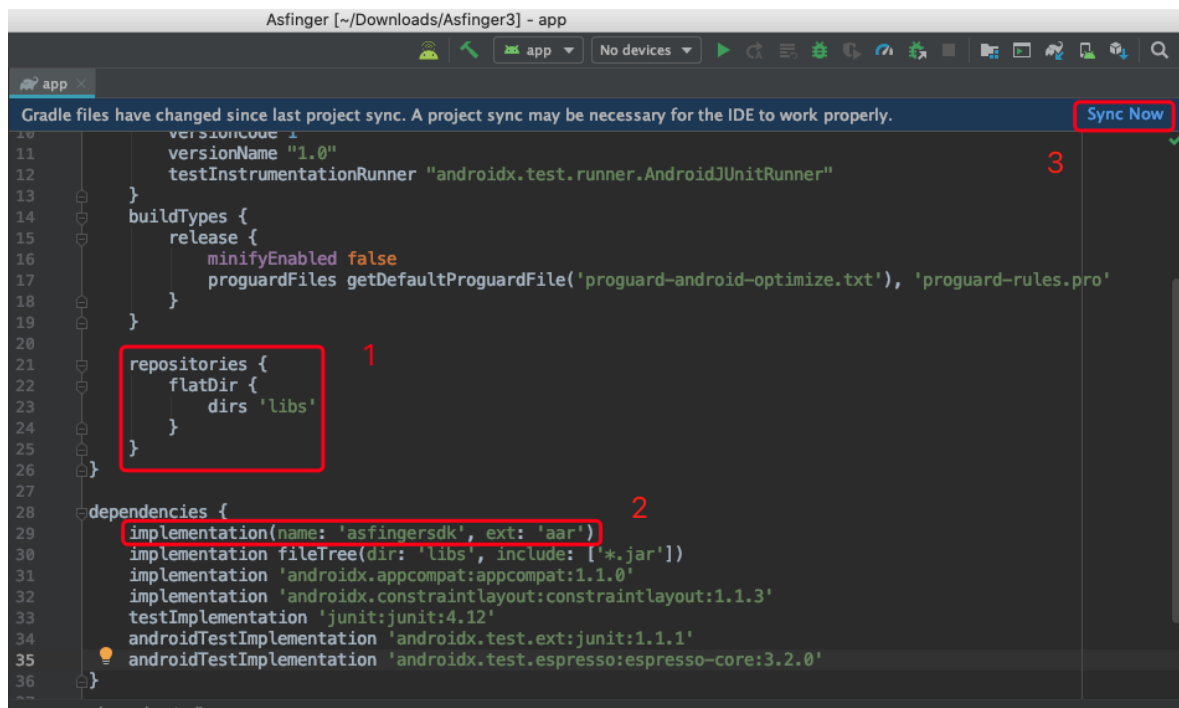


図 1-1-5

成功に同期した場合、下図赤枠のように表示され、SDK のインポートは成功になります。

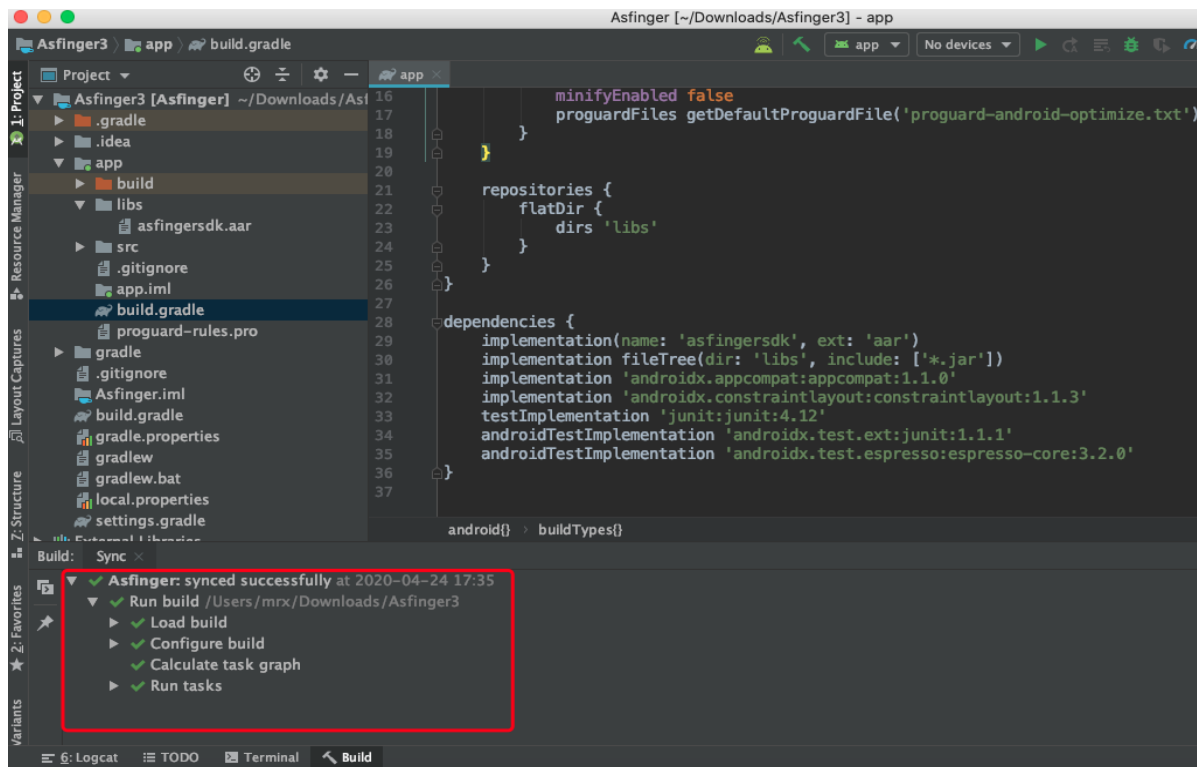


図 1-1-6

1.2 ライブラリーファイルを使う

ライブラリーファイルを使うクラスに、「import」文で AsFingerSDK ライブラリーファイルを引用します（図 1-2-1）。

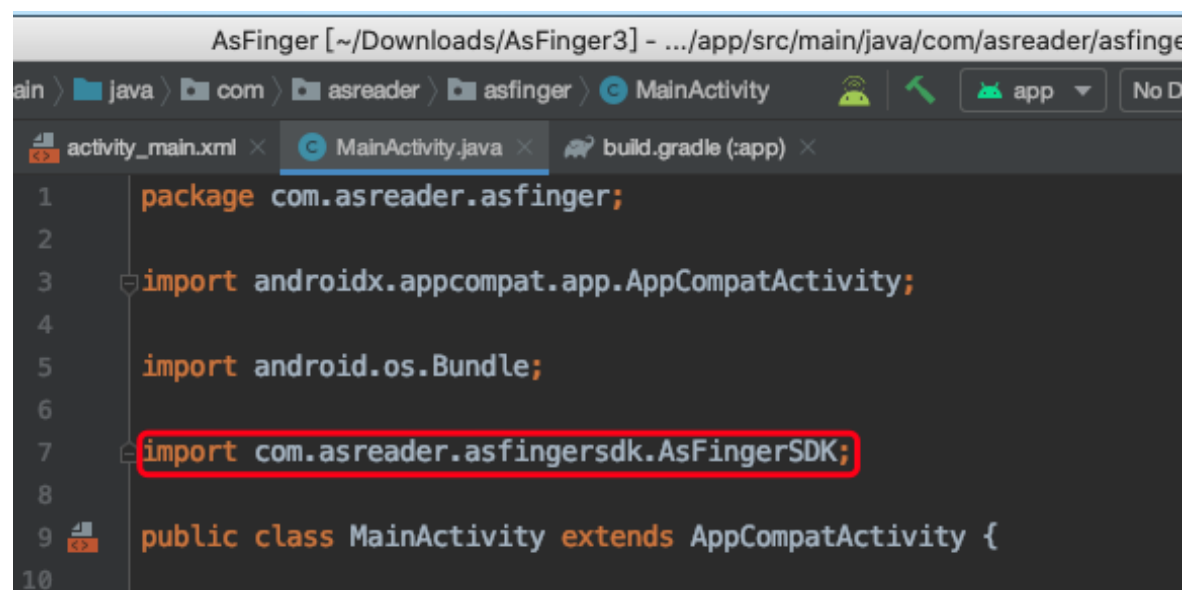


図 1-2-1

下図（図 1-2-2）の通りに操作：

AsFingerSDK オブジェクトを生成・初期化して、Activity オブジェクト（this）を渡します。（マーク 1、2、3）

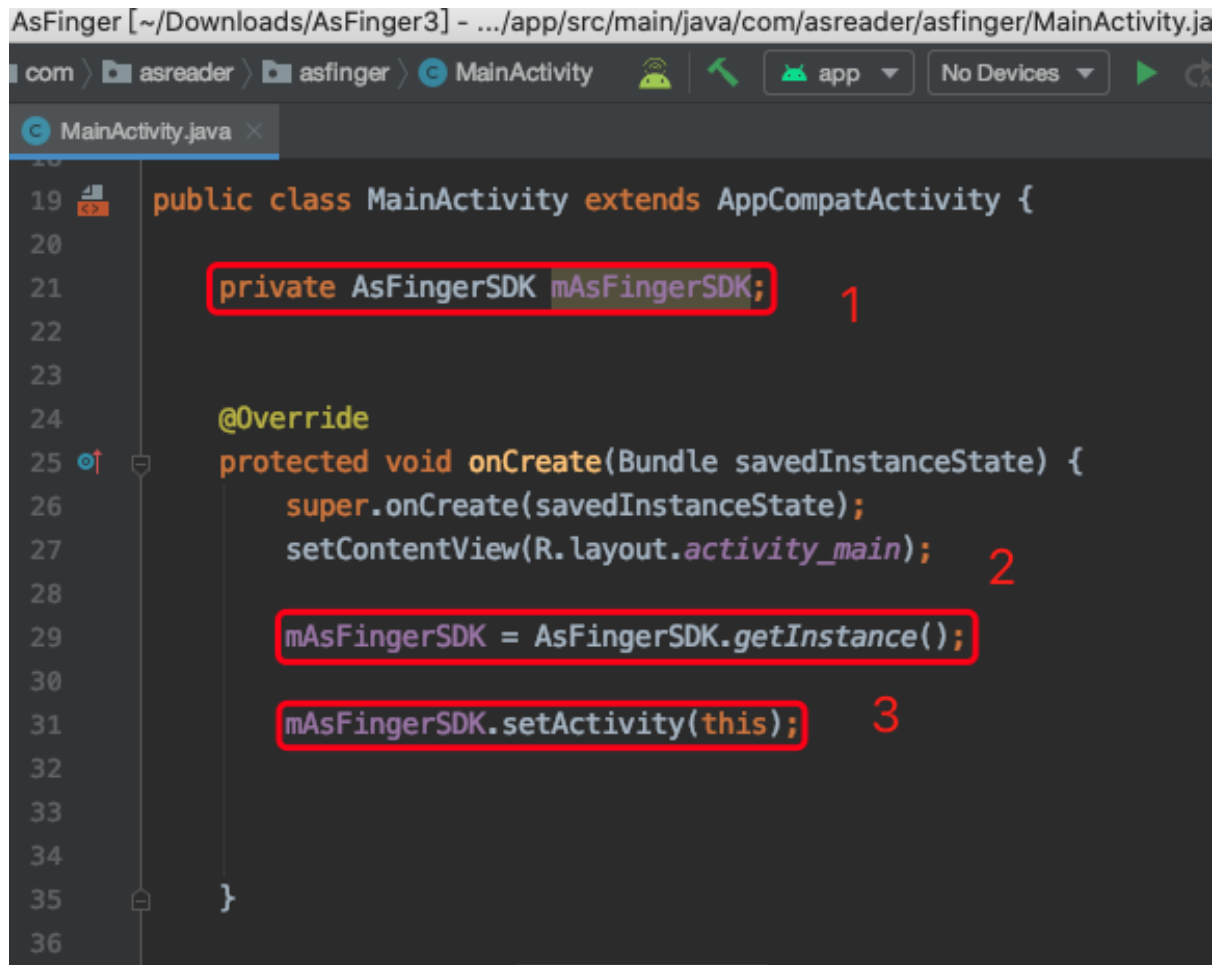


図 1-2-2

API の使い方：「connect(BluetoothDevice device, DeviceType type)」メソッドを例として、下図（図 1-2-3）の通りに操作します。

- ① 「Import」文で「BluetoothAdapter」、「BluetoothDevice」及び DeviceType ライブラリーファイルを引用します。（マーク 1）
- ② iOS デバイスの Bluetooth アダプター「BluetoothAdapter」オブジェクトを取得します。（マーク 3）
- ③ BluetoothDevice オブジェクトを取得するために Bluetooth アドレスを定義します。（マーク 4）
- ④ 「mAsFingerSDK.connect(device, DeviceType.spp)」メソッドを実行で接続します。（マーク 5）

```

13
14 import com.asreader.asfingersdk.AsFingerSDK;
15 import android.bluetooth.BluetoothDevice;
16 import android.bluetooth.BluetoothAdapter;
17 import com.asreader.asfingersdk.type.DeviceType;
18
19 public class MainActivity extends AppCompatActivity {
20
21     private AsFingerSDK mAsFingerSDK;
22     private BluetoothAdapter mBluetoothAdapter;
23
24     @Override
25     protected void onCreate(Bundle savedInstanceState) {
26         super.onCreate(savedInstanceState);
27         setContentView(R.layout.activity_main);
28
29         mAsFingerSDK = AsFingerSDK.getInstance();
30
31         mAsFingerSDK.setActivity(this);
32
33         mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
34
35         String address = "DC:0D:30:60:9C:02";
36         BluetoothDevice device = mBluetoothAdapter.getRemoteDevice(address);
37
38         mAsFingerSDK.connect(device, DeviceType.spp);
39
40
41     }
42
43

```

图 1-2-3

注: AsFingerEventListener オブジェクトを生成・初期化して、AsFingerEventListener インターフェースにあるメソッドを書き換えます。(図 1-2-4)。

```
private AsFingerEventListener mAsFingerEventListener = new AsFingerEventListener() {  
    @Override  
    public void onStateChanged(AsFingerConnectionType asFingerConnectionType) {}  
  
    @Override  
    public void receivedBarcodeData(byte[] bytes) {}  
  
    @Override  
    public void receivedBattery(int i) {}  
  
    @Override  
    public void receivedMacAddress(String s) {}  
  
    @Override  
    public void receivedFirmwareVersion(String s) {}  
  
    @Override  
    public void receivedData(byte[] bytes) {}  
  
    @Override  
    public void receivedDevice(BluetoothDevice bluetoothDevice) {}  
  
    @Override  
    public void receivedFoundDeviceFinished() {}  
};
```

図 1-2-4

注: Bluetooth デバイスを接続する前に、
「setAsFingerEventListener(AsFingerEventListener asFingerEventListener)」メソッドを実行して、AsFingerEventListener オブジェクトを渡すことで、Bluetooth 接続

状態をモニターし、AsReader Finger-Type から返ったデータを受け取ります。（図 1-2-5）

```
public class MainActivity extends AppCompatActivity {  
  
    private AsFingerSDK mAsFingerSDK;  
    private BluetoothAdapter mBluetoothAdapter;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        mAsFingerSDK = AsFingerSDK.getInstance();  
  
        mAsFingerSDK.setActivity(this);  
  
        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();  
  
        String address = "DC:0D:30:60:9C:02";  
        BluetoothDevice device = mBluetoothAdapter.getRemoteDevice(address);  
  
        mAsFingerSDK.setAsFingerEventListener(mAsFingerEventListener);  
  
        mAsFingerSDK.connect(device, DeviceType.spp);  
    }  
}
```

図 1-2-5

注：インターフェースにある「onStateChanged(AsFingerConnectionType connectionType)」と「receivedBarcodeData(byte[] barcodeData)」メソッドを例とします。（図 1-2-6）

```
@Override
public void onStateChanged(BluetoothConnectionType bluetoothConnectionType) {
    switch (bluetoothConnectionType){//Determine the connection status
        case Connected://Connected

            break;
        case Disconnected://Disconnected

            break;
    }
}

@Override
public void receivedBarcodeData(byte[] barcodeData) {
    // Once the function startScan() is called, the device returns the barcode data scanned.
    // Encoding
    // Charset mCharset = Charset.forName("ASCII");
    // Charset mCharset = Charset.forName("Shift_JIS");
    // Charset mCharset= Charset.forName("UTF-8");
    // String str = "";
    // try {
    //     str = new String(barcodeData, mCharset);
    // } catch (Exception e) {}
    // str = "6921734976727"
}
```

図 1-2-6

1.3 プロジェクト権限を追加

SDK を使用する前にアプリケーションの AndroidManifest.xml ファイルにプロジェクト権限を定義する必要があります（図 1-2-7）：

1. アプリケーションからはペアリングする Bluetooth デバイスへの接続を許可します。（マーク 1）
2. アプリケーションからは Bluetooth デバイスを検索し、ペアリングすることを許可します。（マーク 2）
3. アプリケーションが精確な位置へのアクセスを許可します。（アプリケーションが Bluetooth デバイスの検索機能を追加する場合にのみ許可）（マーク 3）
4. アプリケーションが大体な位置へのアクセスを許可します。（アプリケーションが Bluetooth デバイスの検索機能を追加する場合にのみ許可）（マーク 4）
5. アプリケーションから内蔵の SD カードへデータの送信を許可します。（マーク 5）
6. アプリケーションが内蔵の SD カードからデータの取得を許可します。（マーク 6）

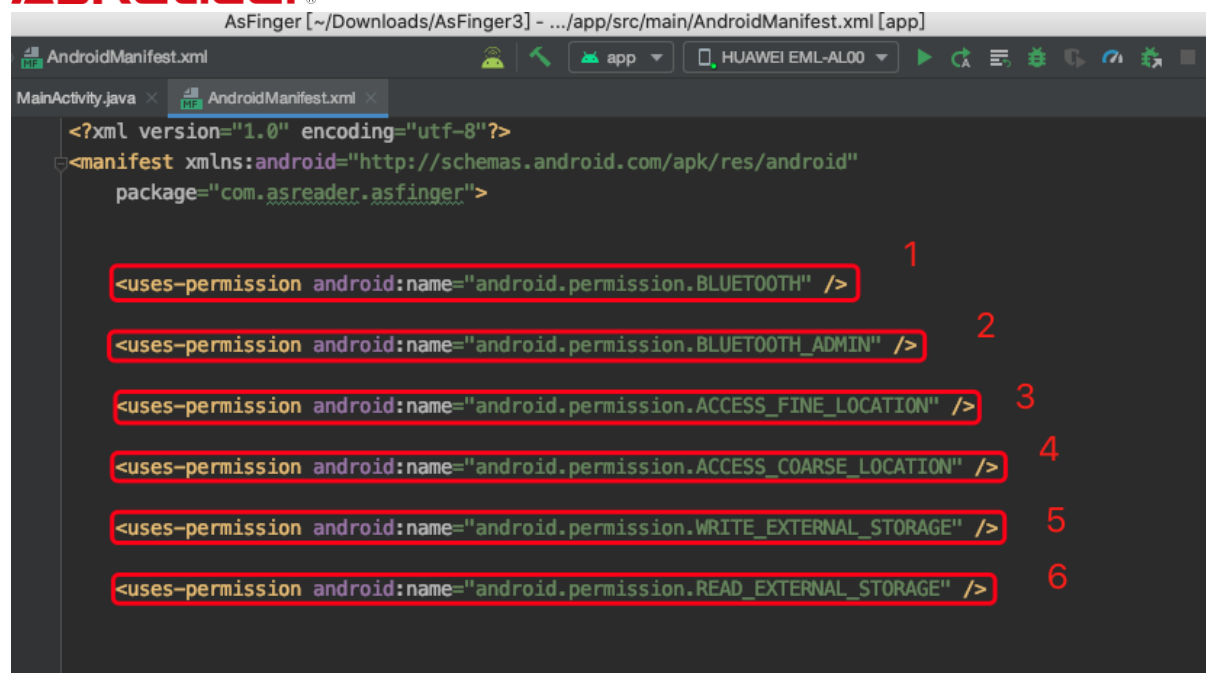


图 1-2-7

2 メソッドの説明

2.1 AsFingerSDK

2.1.1 getInstance

関数名	AsFingerSDK getInstance();		
引数名	IN/OUT	型	説明
	OUT	AsFingerSDK	AsFingerSDK
■メソッドの説明： AsFingerSDK オブジェクトを取得します。 ■サンプルコード： <pre>//AsFingerSDK オブジェクトを生成・初期化します。 AsFingerSDK mAsFingerSDK = AsFingerSDK.getInstance();</pre>			

2.1.2 setAsFingerEventListener

関数名	void setAsFingerEventListener(AsFingerEventListener asFingerEventListener);		
引数名	IN/OUT	型	説明
asFingerEvent Listener	IN	AsFingerEventListener	AsFingerEventListener オブジェクト (2.2 をご参照)。

■メソッドの説明：

AsFingerEventListener のモニターを設定します。

「connect(BluetoothDevice device, DeviceType type)」メソッドを実行する前に実行する必要があります。

■サンプルコード：

//AsFingerSDK オブジェクトを生成・初期化します。

AsFingerSDK mAsFingerSDK = AsFingerSDK.getInstance();

//this は Activity 型です。

mAsFingerSDK.setActivity(this);

/AsFingerEventListener をインスタンス化します。

```
private AsFingerEventListener mAsFingerEventListener = new AsFingerEventListener() {
```

```
    @Override
```

```
    public void onStateChanged(AsFingerConnectionType connectionType) {
```

```
        具体的な実現は 2.2.1 をご参照
```

```
    }
```

```
}
```

```
    @Override
```

```
    public void receivedBarcodeData(byte[] barcodeData) {
```

```
        具体的な実現は 2.2.2 をご参照
```

```
}
```

```
    @Override
```

```
    public void receivedBattery(int battery) {
```

```
        具体的な実現は 2.2.3 をご参照
```

```
}
```

```
    @Override
```

```
    public void receivedFirmwareVersion(String firmwareVersion) {
```

```
        具体的な実現は 2.2.4 をご参照
```

```
}
```

```
    @Override
```

```
    public void receivedData(byte[] data) {
```

```
        具体的な実現は 2.2.5 をご参照
```

```
}
```

```
    @Override
```

```
    public void receivedDevice(BluetoothDevice bluetoothDevice) {
```

```
        具体的な実現は 2.2.6 をご参照
```

```
}
```

```
    @Override
```

```
    public void receivedFoundDeviceFinished() {
```

```
        具体的な実現は 2.2.7 をご参照
```

```
}
```

```
};
```

//AsFingerEventListener のモニターを設定します。

mAsFingerSDK.setAsFingerEventListener(mAsFingerEventListener);

2.1.3 connect

関数名	void connect(BluetoothDevice device, DeviceType type);		
引数名	IN/OUT	型	説明
device	IN	BluetoothDevice	Bluetooth オブジェクト
type	IN	DeviceType	接続モード、列挙型 (3.1 をご参照)
■メソッドの説明： Bluetooth を接続します。（AsReader Finger-Type） 当該メソッドを実行後に「onStateChanged(AsFingerConnectionType connectionType);」メソッドで AsReader Finger-Type の接続状態をコールバックします。（2.2.1 をご参照）。 ■サンプルコード： <pre>//AsFingerSDK オブジェクトを生成・初期化します。 AsFingerSDK mAsFingerSDK = AsFingerSDK.getInstance(); // this は Activity 型です。 mAsFingerSDK.setActivity(this); //iOS デバイスの Bluetooth アダプターを取得します。 BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter(); // AsReader Finger-Type の Bluetooth アドレスをオブジェクトを生成・初期化します。 String address = "DC:0D:30:60:9C:02"; //BluetoothDevice オブジェクトの device を取得します。 BluetoothDevice device = mBluetoothAdapter.getRemoteDevice(address); //connect(BluetoothDevice device, DeviceType type);メソッドを実行して、AsReader Finger-Type を //接続します。 mAsFingerSDK.connect(device, DeviceType.spp);</pre>			

2.1.4 setActivity

関数名	void setActivity(Activity activity)		
引数名	IN/OUT	型	説明
activity	IN	Activity	環境
■メソッドの説明： アプリケーションのライフサイクルと同じ環境を設定します。（例：MainActivity） ■サンプルコード： <pre>// AsFingerSDK オブジェクトを生成・初期化します。 AsFingerSDK mAsFingerSDK = AsFingerSDK.getInstance(); // this は Activity 型です。 mAsFingerSDK.setActivity (this);</pre>			

2.1.5 disconnect

関数名	void disconnectDevice();		
引数名	IN/OUT	型	説明
■メソッドの説明： Bluetooth を切断します。（AsReader Finger-Type） 当該メソッドを実行後に「onStateChanged(AsFingerConnectionType connectionType);」メソッドで AsReader Finger-Type の接続状態をコールバックします。（2.2.1 をご参照）。 ■サンプルコード： <pre>// AsFingerSDK オブジェクトを生成・初期化します。</pre>			

```
AsFingerSDK mAsFingerSDK = AsFingerSDK.getInstance();
// this は Activity 型です。
mAsFingerSDK.setActivity(this);
// Bluetooth を切断します。
mAsFingerSDK.disconnect();
```

2.1.6 getSdkVersion

関数名	String getSdkVersion();		
引数名	IN/OUT	型	説明
	OUT	String	SDK バージョン
■メソッドの説明： AsFingerSDK バージョンを取得します。 ■サンプルコード： <pre>// AsFingerSDK オブジェクトを生成・初期化します。 AsFingerSDK mAsFingerSDK = AsFingerSDK.getInstance(); // this は Activity 型です。 mAsFingerSDK.setActivity(this); // sdkVersion は AsFingerSDK バージョンです。 String sdkVersion= mAsFingerSDK.getSdkVersion();</pre>			

2.1.7 getFirmwareVersion

関数名	void getFirmwareVersion();		
引数名	IN/OUT	型	説明
■メソッドの説明： AsReader Finger-Type のファームウェアバージョンを取得します。 当該メソッドを実行後に「receivedFirmwareVersion(String firmwareVersion);」メソッドで AsReader Finger-Type のファームウェアバージョンをコールバックします。 (2.2.4 をご参照)。 ■サンプルコード： <pre>// AsFingerSDK オブジェクトを生成・初期化します。 AsFingerSDK mAsFingerSDK = AsFingerSDK.getInstance(); // this は Activity 型です。 mAsFingerSDK.setActivity(this); // firmwareVersion は AsReader Finger-Type のファームウェアバージョン String firmwareVersion= mAsFingerSDK.getFirmwareVersion ();</pre>			

2.1.8 getBattery

関数名	void getBattery();		
引数名	IN/OUT	型	説明

■メソッドの説明：

AsReader Finger-Type の電池残量を取得します。

当該メソッドを実行後に「receivedBattery(int battery);」メソッドで AsReader Finger-Type の電池残量をコールバックします。（[2.2.3](#)をご参照）。

■サンプルコード：

// AsFingerSDK オブジェクトを生成・初期化します。

AsFingerSDK mAsFingerSDK = AsFingerSDK.getInstance();

// this は Activity 型です。

mAsFingerSDK.setActivity(this);

//AsReader Finger-Type の電池残量を取得します。

mAsFingerSDK.getBattery();

2.1.9 startScan

関数名	void startScan();		
引数名	IN/OUT	型	説明
■メソッドの説明： AsReader Finger-Type はスキャンを開始します。 当該メソッドを実行後に「receivedBarcodeData(byte[] barcodeData);」メソッドで AsReader Finger-Type がスキャンしたデータをコールバックします。（2.2.2をご参照）。 ■サンプルコード： // AsFingerSDK オブジェクトを生成・初期化します。 AsFingerSDK mAsFingerSDK = AsFingerSDK.getInstance(); // this は Activity 型です。 mAsFingerSDK.setActivity(this); //AsReader Finger-Type はスキャンを開始します。 mAsFingerSDK.startScan ();			

2.1.10 stopScan

関数名	void stopScan();		
引数名	IN/OUT	型	説明
■メソッドの説明： AsReader Finger-Type はスキャンを停止します。 ■サンプルコード： // AsFingerSDK オブジェクトを生成・初期化します。 AsFingerSDK mAsFingerSDK = AsFingerSDK.getInstance(); // this は Activity 型です。 mAsFingerSDK.setActivity(this); //AsReader Finger-Type はスキャンを停止します。 mAsFingerSDK.stopScan();			

2.1.11 sendData

関数名	void sendData(byte[] data);		
引数名	IN/OUT	型	説明
data	IN	byte[]	コマンドデータ

■メソッドの説明：

AsReader Finger-Type にコマンドを送信します。

当該メソッドを実行後に「receivedData(byte[] data)」メソッドで AsReader Finger-Type から返ったデータをコールバックします。 ([2.2.5](#) をご参照)。

■サンプルコード：

// AsFingerSDK オブジェクトを生成・初期化します。

AsFingerSDK mAsFingerSDK = AsFingerSDK.getInstance();

// this は Activity 型です。

mAsFingerSDK.setActivity(this);

//byte[] タイプのコマンドをオブジェクトを生成・初期化します。

byte[] data = new byte[]{0x03};

// AsReader Finger-Type にコマンドを送信します。

mAsFingerSDK.sendData(data);

2.1.12 startDiscovery

関数名	void startDiscovery();		
引数名	IN/OUT	型	説明
■メソッドの説明： Bluetooth デバイスを検索します。 当該メソッドを実行後に「receivedDevice(BluetoothDevice device)」メソッドで検索できた Bluetooth デバイスをコールバックします。 (2.2.6 をご参照)。 ■サンプルコード： // AsFingerSDK オブジェクトを生成・初期化します。 AsFingerSDK mAsFingerSDK = AsFingerSDK.getInstance(); // this は Activity 型です。 mAsFingerSDK.setActivity(this); // Bluetooth デバイスを検索します。 mAsFingerSDK.startDiscovery();			

2.1.13 stopDiscovery

関数名	void stopDiscovery();		
引数名	IN/OUT	型	説明
■メソッドの説明： Bluetooth デバイスの検索を停止します。 当該メソッドを実行後に「receivedFoundDeviceFinished()」メソッドで Bluetooth デバイスの検索停止情報をコールバックします。 (2.2.7 をご参照)。 ■サンプルコード： // AsFingerSDK オブジェクトを生成・初期化します。 AsFingerSDK mAsFingerSDK = AsFingerSDK.getInstance(); // this は Activity 型です。 mAsFingerSDK.setActivity(this); Bluetooth デバイスの検索を停止します。 mAsFingerSDK.stopDiscovery();			

2.1.14 getPairedDevices

関数名	Set<BluetoothDevice> getPairedDevices();
-----	--

引数名	IN/OUT	型	説明
	OUT	Set<Bluetooth Device>	Bluetooth デバイスのリスト
■メソッドの説明： ペアリングしている Bluetooth デバイスのリストを取得します。 ■サンプルコード： // AsFingerSDK オブジェクトを生成・初期化します。 AsFingerSDK mAsFingerSDK = AsFingerSDK.getInstance(); // this は Activity 型です。 mAsFingerSDK.setActivity(this); //ペアリングしている Bluetooth デバイスのリスト Set<BluetoothDevice> pairedDevices = mAsFingerSDK. getPairedDevices();			

2.1.15 deviceType

関数名	DeviceType deviceType();		
引数名	IN/OUT	型	説明
	OUT	DeviceType	接続モード、列挙型 (3.1 をご参照)。
■メソッドの説明： 接続している AsReader Finger-Type の Bluetooth 接続モードを取得します。 ■サンプルコード： // AsFingerSDK オブジェクトを生成・初期化します。 AsFingerSDK mAsFingerSDK = AsFingerSDK.getInstance(); // this は Activity 型です。 mAsFingerSDK.setActivity(this); //接続している Bluetooth 接続モード DeviceType mDeviceType = mAsFingerSDK.deviceType();			

2.1.16 connectedDevice

関数名	BluetoothDevice connectedDevice();		
引数名	IN/OUT	型	説明
	OUT	BluetoothDevice	Bluetooth デバイス
■メソッドの説明： 接続している Bluetooth デバイスを取得します。 ■サンプルコード： // AsFingerSDK オブジェクトを生成・初期化します。 AsFingerSDK mAsFingerSDK = AsFingerSDK.getInstance(); // this は Activity 型です。 mAsFingerSDK.setActivity(this); //接続している Bluetooth デバイスを取得します。 BluetoothDevice mBluetoothDevice = mAsFingerSDK.connectedDevice();			

2.1.17 autoConnect

関数名	boolean autoConnect();		
引数名	IN/OUT	型	説明
	OUT	boolean	true : 自動接続 ON false : 自動接続 OFF

■メソッドの説明：

Android デバイスが AsReader Finger-Type へ自動的に接続するかどうかの設定を取得します。

■サンプルコード：

```
// AsFingerSDK オブジェクトを生成・初期化します。
AsFingerSDK mAsFingerSDK = AsFingerSDK.getInstance();
// this は Activity 型です。
mAsFingerSDK.setActivity(this);
// Android デバイスから AsReader Finger-Type へ自動的に接続するかどうかの設定を取得します。
boolean isAutoConnect = mAsFingerSDK.autoConnect();
```

2.1.18 setAutoConnect

関数名	setAutoConnect(boolean isAutoConnect);		
引数名	IN/OUT	型	説明
isAutoConnect	IN	boolean	true : 自動接続 ON false : 自動接続 OFF

■メソッドの説明：

Android デバイスが AsReader Finger-Type へ自動的に接続するかどうかを設定します。

■サンプルコード：

```
// AsFingerSDK オブジェクトを生成・初期化します。
AsFingerSDK mAsFingerSDK = AsFingerSDK.getInstance();
// this は Activity 型です。
mAsFingerSDK.setActivity(this);
// Android デバイスが AsReader Finger-Type へ自動的に接続するかどうかを設定します。
mAsFingerSDK.setAutoConnect(true);
```

2.2 AsFingerEventListener

2.2.1 onStateChanged

関数名	void onStateChanged(AsFingerConnectionType connectionType);		
引数名	IN/OUT	型	説明
connectionType	IN	AsFingerConnectionType	Bluetooth 接続状態、列挙型 (3.2 をご参照)。

■メソッドの説明：

AsReader Finger-Type の接続状態モニターします。

「connect(BluetoothDevice device, DeviceType type);」、「disconnect();」メソッドを実行する時、または AsReader Finger-Type 自身の原因で接続が切断された時に、コールバックされます ([2.1.3](#) をご参照)、または ([2.1.5](#) をご参照)。

■サンプルコード：

```
@Override
public void onStateChanged(AsFingerConnectionType connectionType) {
    //Determine the connection status
    switch (connectionType){
//Connected
        case Connected:

            break;
//Disconnected
        case Disconnected:

            break;
    }
}
```

2.2.2 receivedBarcodeData

関数名	void receivedBarcodeData(byte[] barcodeData);		
引数名	IN/OUT	型	説明
barcodeData	IN	byte[]	スキャンしたバーコードデータ

■メソッドの説明：

AsReader Finger-Type がスキャンしたデータを受け取ります。

「startScan();」メソッドを実行して、データをスキャンできた時にコールバックされます。 ([2.1.9](#) をご参照)。

■サンプルコード：

```
@Override
public void receivedBarcodeData(byte[] barcodeData) {
    // Once the function startScan() is called, the device returns the barcode data //scanned.
    // Encoding
    // Charset mCharset = Charset.forName("ASCII");
    // Charset mCharset = Charset.forName("Shift_JIS");
    // Charset mCharset= Charset.forName("UTF-8");
    // String str = "";
    // try {
    // str = new String(barcodeData, mCharset);
    //} catch (Exception e) {}
    //返ったデータ例 str = "6921734976727"
```

```
}
```

2.2.3 receivedBattery

関数名	void receivedBattery(int battery);		
引数名	IN/OUT	型	説明
battery	IN	Int	電池残量 値の範囲 : : 1-100。
■メソッドの説明 : AsReader Finger-Type の電池残量を受け取ります。 「getBattery();」メソッドを実行する時に、コールバックされます。(2.1.8をご参照)。 ■サンプルコード : <pre>@Override public void receivedBattery(int battery) { // Once the function getBattery() is called, the device returns the data for the // battery's remaining. // 返ったデータ例 battery = 60 }</pre>			

2.2.4 receivedFirmwareVersion

関数名	void receivedFirmwareVersion (String firmwareVersion);		
引数名	IN/OUT	型	説明
firmwareVersion	IN	String	ファームウェアバージョン
■メソッドの説明 : AsReader Finger-Type のファームウェアバージョンを受け取ります。 「getFirmwareVersion();」メソッドを実行する時に、コールバックされます。(2.1.7をご参照)。 ■サンプルコード : <pre>@Override public void receivedFirmwareVersion(String firmwareVersion) { // Once the function getFirmwareVersion() is called, the device returns the data //for thefirmware version of the device. // 返ったデータ例 firmwareVersion = "FW_1.0.0" }</pre>			

2.2.5 receivedData

関数名	void receivedData(byte[] data);		
引数名	IN/OUT	型	説明

data	IN	byte[]	データ
■メソッドの説明： AsReader Finger-Type から返ったデータを受け取ります。 「sendData(byte[] data);」、「getFirmwareVersion();」、「getBattery();」または「startScan();」メソッドを実行して、バーコードをスキャンできた時に、コールバックされます。（2.1.11をご参照）、（2.1.7をご参照）、（2.1.8をご参照）または（2.1.9をご参照）。 ■サンプルコード： <pre> @Override public void receivedData(byte[] data) { // Once the function One of all the interfaces to send commands is called or //sendData(byte[] data) is called the device returns // the Corresponding data // Encoding // Charset mCharset = Charset.forName("ASCII"); // Charset mCharset = Charset.forName("Shift_JIS"); // Charset mCharset= Charset.forName("UTF-8"); // String str = ""; // try { // str = new String(data, mCharset); //} catch (Exception e) {} //返ったデータ例 str = "6921734976727" } }; </pre>			

2.2.6 receivedDevice

関数名	void receivedDevice(BluetoothDevice bluetoothDevice);		
引数名	IN/OUT	型	説明
bluetoothDevice	IN	BluetoothDevice	Bluetooth デバイス
■メソッドの説明： 検索できた Bluetooth デバイスを受け取ります。 「startDiscovery()」メソッドを実行して、Bluetooth デバイスを検索できた時に、コールバックされます。（2.1.12をご参照）。 ■サンプルコード： <pre> @Override public void receivedDevice(BluetoothDevice bluetoothDevice) { // Once the function One of all the interfaces to send commands is called or // startDiscovery () is called the device returns BluetoothDevice // bluetoothDevice は検索できた Bluetooth デバイスです。 }); } </pre>			

2.2.7 receivedFoundDeviceFinished

関数名	void receivedFoundDeviceFinished();		
引数名	IN/OUT	型	説明

■メソッドの説明：

Bluetooth デバイスの検索停止情報を受け取ります。

「stopDiscovery()」メソッドを実行して、Bluetooth デバイスの検索を停止する時に、コールバックされます。（[2.1.13](#)をご参照）。

■サンプルコード：

```
@Override
public void receivedFoundDeviceFinished(){
    // Once the function One of all the interfaces to send commands is called or
    //stopDiscovery () is called
};
```

3 Enum

3.1 DeviceType

定義	説明
unknow = 0	no connection
ble = 1	ble connection
spp = 2	spp connetion

3.2 AsFingerConnectionType

定義	説明
connected = 0	connected
connecting = 1	connecting
disconnected = 2	disconnected