AsReaderP252B SDK

Android SDK Reference Guide

Asterisk Inc.

Revision History

Version	Description	Date
V1.0	Initial version	2024/3/12

Contents

Introduction

This manual provides the following information to developers developing Android applications using the SDK.

- How the development environment is built.
- Description of various SDK library functions.

Development tools:

- Android Studio Arctic Fox | 2020.3.1
- Android SDK 24
- Android Gradle 8.1

System requirements:

- Android 10.0+

1 Preparation for SDK Usage

1.1 Import SDK

1. Click the project file "libs" in the app folder and right-click "Open in → Finder" (FIG. 1-1-1).

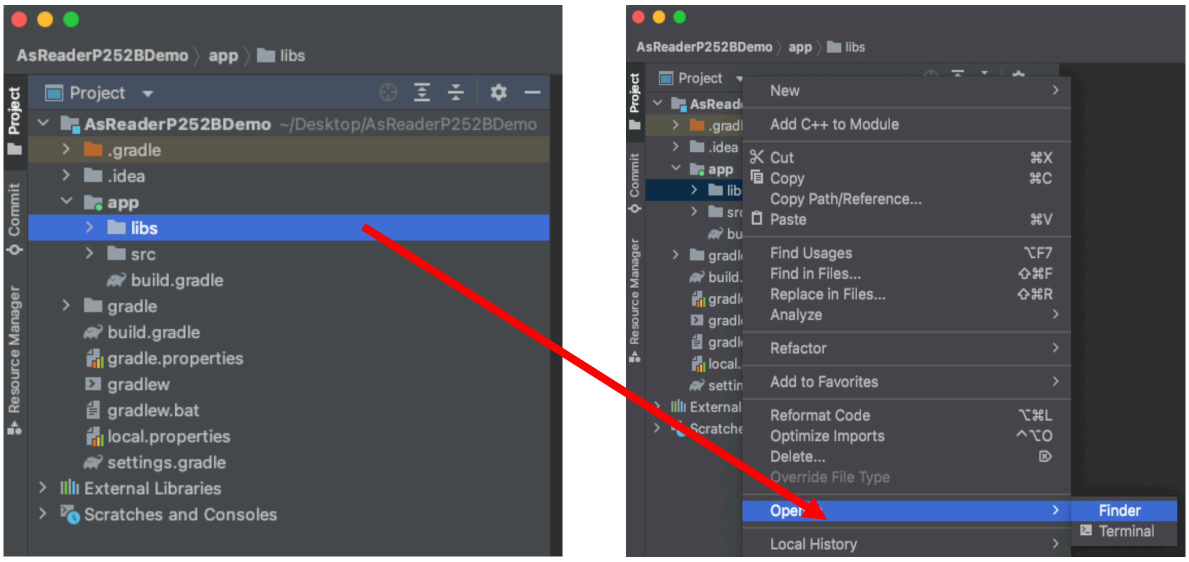


FIG. 1-1-1

2. Select " AsReaderP252BSDK.aar " (FIG. 1-1-2) and " AsReaderP252BSDK.aar" will appear under "libs" of this project (FIG. 1-1-3).

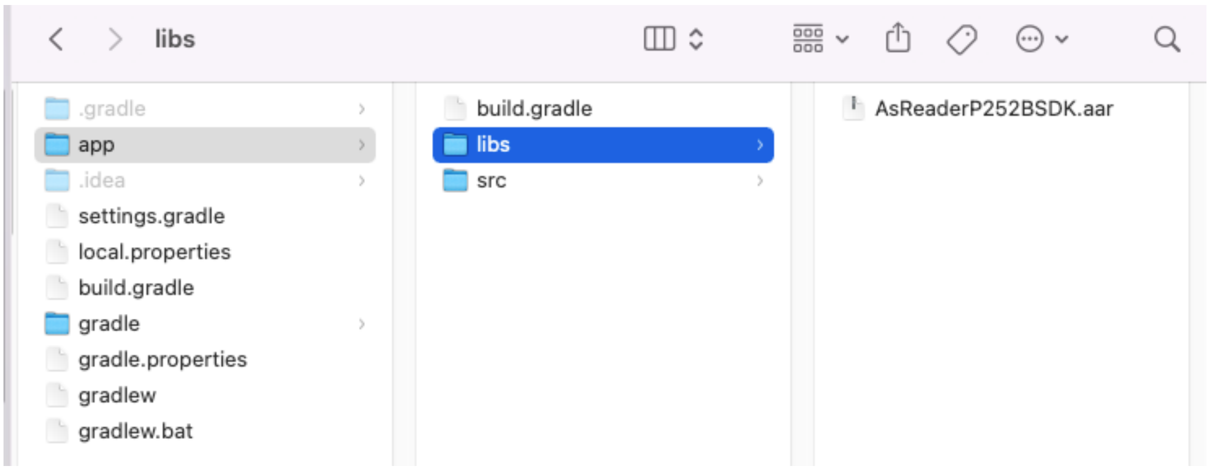


FIG. 1-1-2

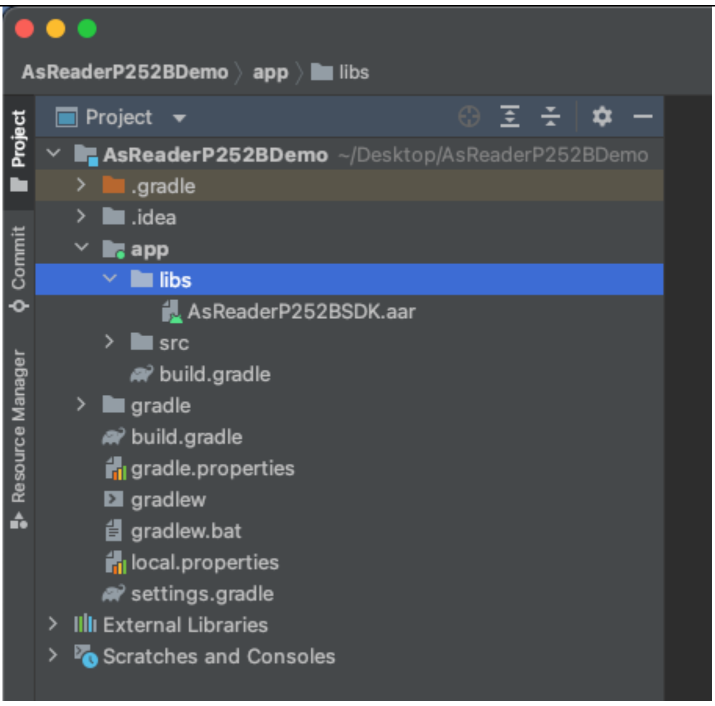


FIG. 1-1-3

3. Double-click to open "build.gradle" in the project (FIG. 1-1-4).

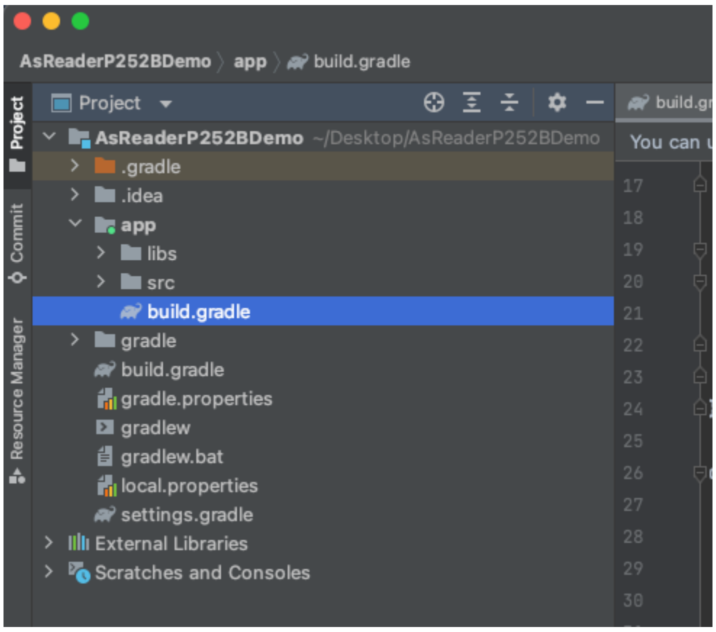
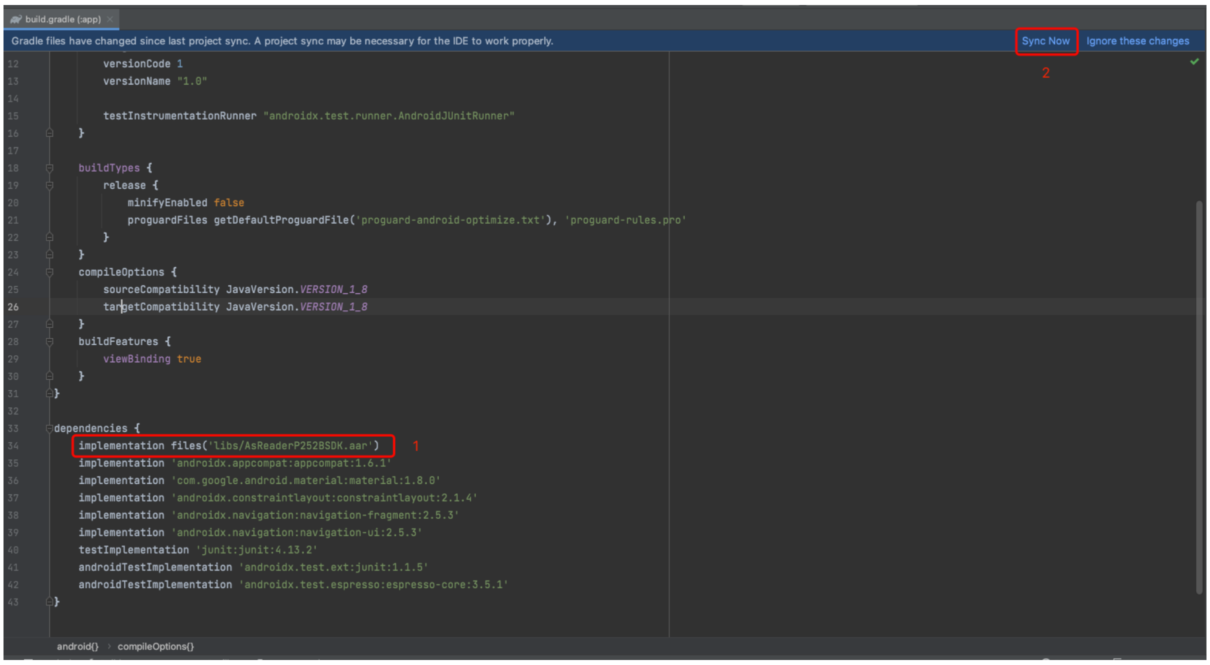


FIG. 1-1-4

4. Add the repositories and dependencies and then click “Sync Now” (FIG.1-1-5).

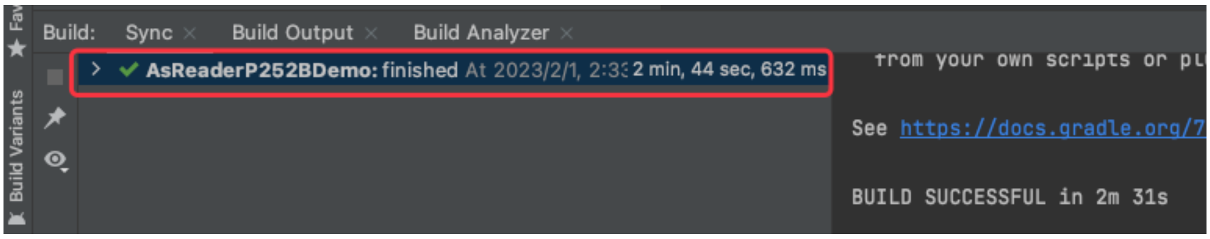


Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

```
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    buildFeatures {
        viewBinding true
    }
}

dependencies {
    implementation files('libs/AsReaderP252BSDK.aar')
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.8.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    implementation 'androidx.navigation:navigation-fragment:2.5.3'
    implementation 'androidx.navigation:navigation-ui:2.5.3'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
}
```

FIG. 1-1-5

5. Successful synchronization is indicated as shown in the FIG.1-1-6. At this point, SDK import is complete.



```
AsReaderP252BDemo: finished At 2023/2/1, 2:3 2 min, 44 sec, 632 ms
```

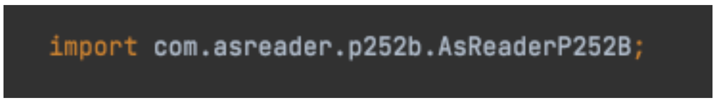
```
from your own scripts or pl

See https://docs.gradle.org/7

BUILD SUCCESSFUL in 2m 31s
```

FIG. 1-1-6

1.2 SDK Usage

1.2.1 Import the SDK

In the class to use the SDK, use the "import" statement to reference the library files (FIG. 1-2-1).

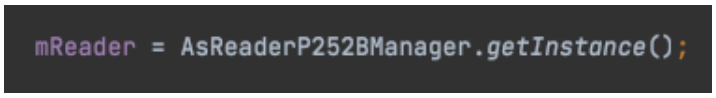


```
import com.asreader.p252b.AsReaderP252B;
```

FIG. 1-2-1

1.2.2 Create and initialize the AsReaderP252B object

Obtain the AsReaderP252B object from AsReaderP252Bmanager and call the function asreaderP252Bmanager.getInstance() to create the AsReaderP252B object (FIG. 1-2-2).

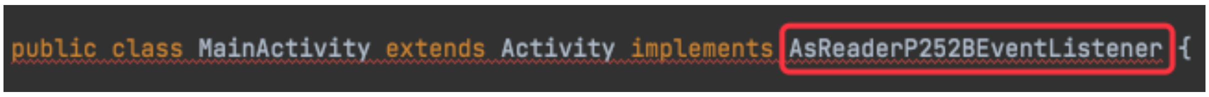


```
mReader = AsReaderP252BManager.getInstance();
```

FIG. 1-2-2

1.2.3 Implement AsReaderP252BEventListener

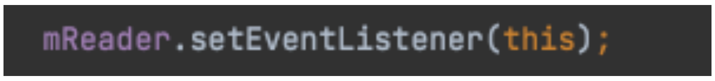
1. Implement AsReaderP252BEventListener (FIG. 1-2-3).



```
public class MainActivity extends Activity implements AsReaderP252BEventListener {
```

FIG. 1-2-3

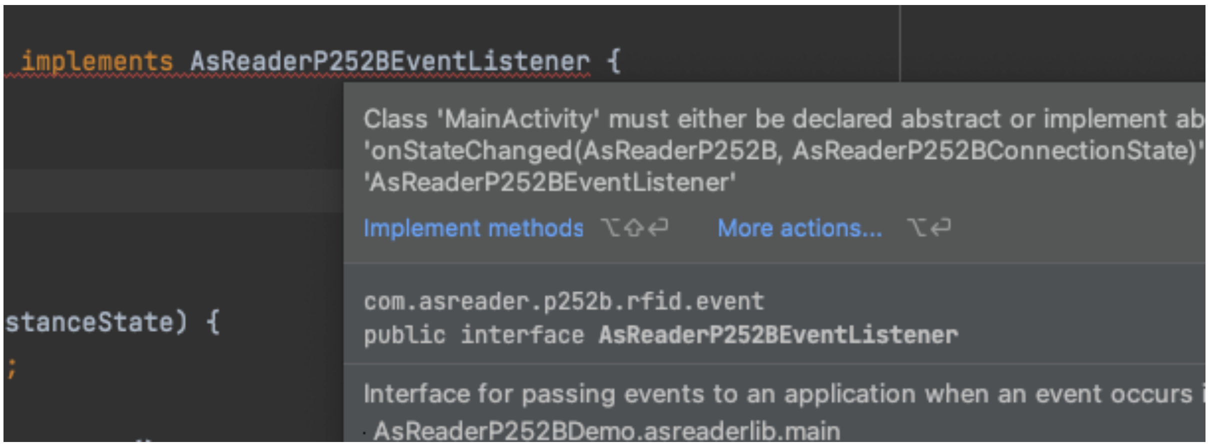
2. Set listener (FIG. 1-2-4).



```
mReader.setEventListener(this);
```

FIG. 1-2-4

3. Move the cursor to the implemented AsReaderP252BEventListener and click the "Implement methods" (FIG. 1-2-5).



```
implements AsReaderP252BEventListener {

stanceState) {
;
```

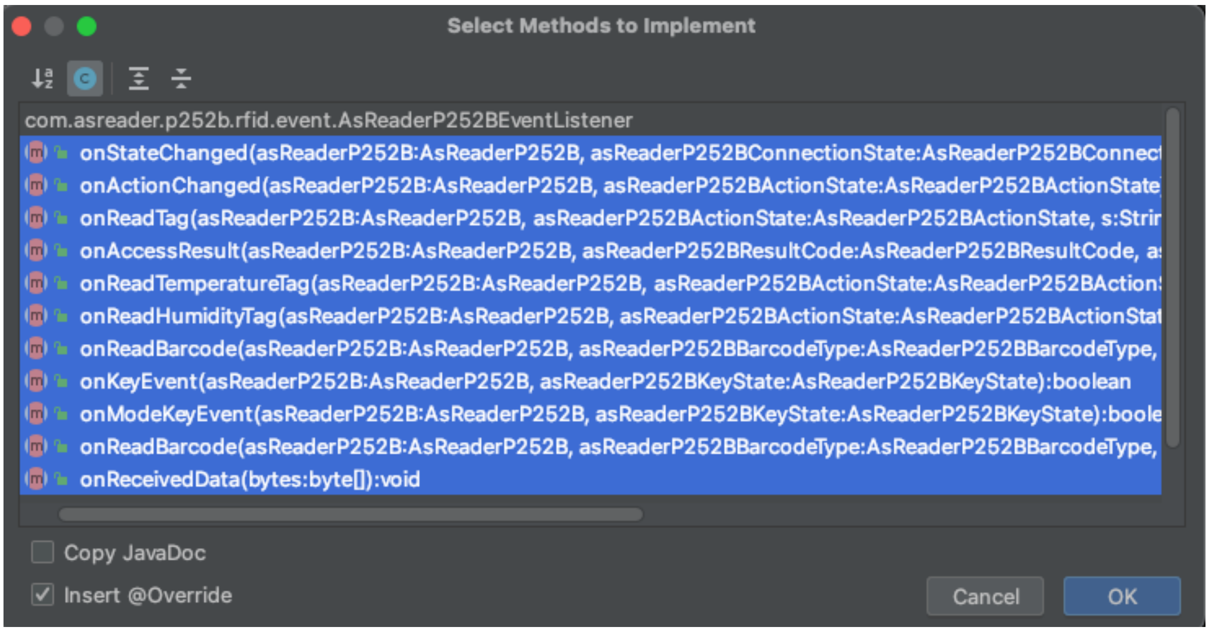
Class 'MainActivity' must either be declared abstract or implement ab
'onStateChanged(AsReaderP252B, AsReaderP252BConnectionState)'
'AsReaderP252BEventListener'

```
com.asreader.p252b.rfid.event
public interface AsReaderP252BEventListener
```

Interface for passing events to an application when an event occurs i
AsReaderP252BDemo.asreaderlib.main

FIG. 1-2-5

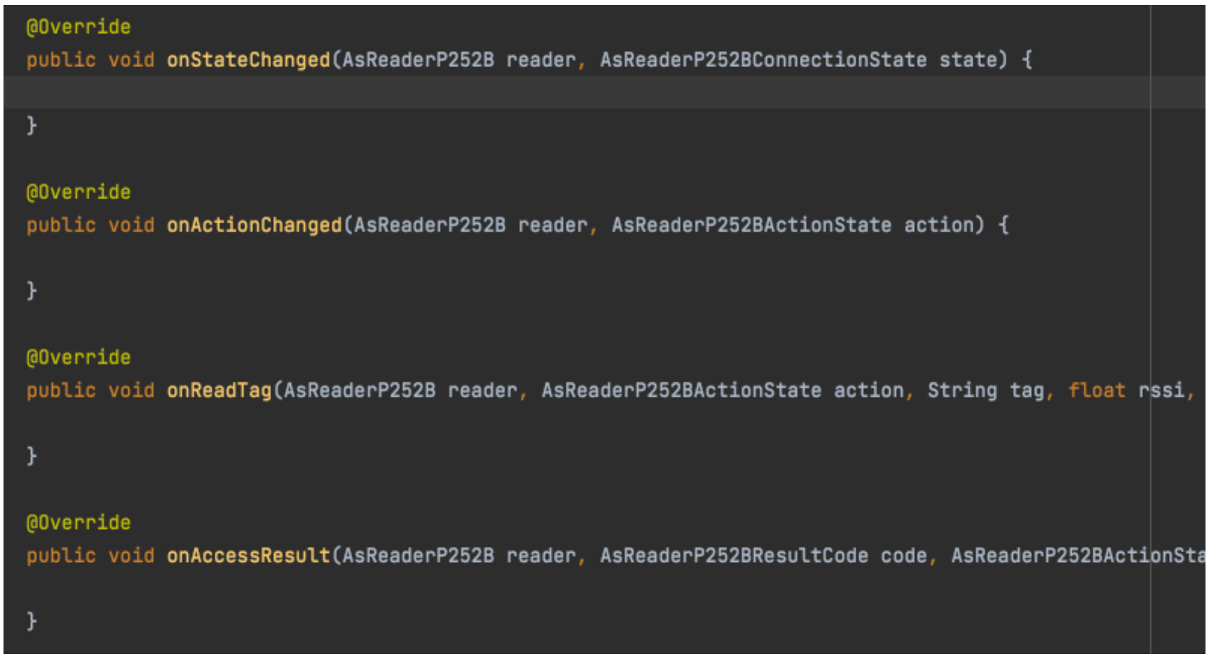
4. Select the functions, and click the OK button (FIG. 1-2-6).



```
com.asreader.p252b.rfid.event.AsReaderP252BEventListener
onStateChanged(asReaderP252B:AsReaderP252B, asReaderP252BConnectionState:AsReaderP252BConnect
onActionChanged(asReaderP252B:AsReaderP252B, asReaderP252BActionState:AsReaderP252BActionState
onReadTag(asReaderP252B:AsReaderP252B, asReaderP252BActionState:AsReaderP252BActionState, s:Strir
onAccessResult(asReaderP252B:AsReaderP252B, asReaderP252BResultCode:AsReaderP252BResultCode, a
onReadTemperatureTag(asReaderP252B:AsReaderP252B, asReaderP252BActionState:AsReaderP252BAction
onReadHumidityTag(asReaderP252B:AsReaderP252B, asReaderP252BActionState:AsReaderP252BActionStat
onReadBarcode(asReaderP252B:AsReaderP252B, asReaderP252BBarcodeType:AsReaderP252BBarcodeType,
onKeyEvent(asReaderP252B:AsReaderP252B, asReaderP252BKeyState:AsReaderP252BKeyState):boolean
onModeKeyEvent(asReaderP252B:AsReaderP252B, asReaderP252BKeyState:AsReaderP252BKeyState):boole
onReadBarcode(asReaderP252B:AsReaderP252B, asReaderP252BBarcodeType:AsReaderP252BBarcodeType,
onReceivedData(bytes:byte[]):void
```

FIG. 1-2-6

5. When the OK button is clicked, the functions in the AsReaderP252BEventListener are automatically created (FIG. 1-2-7).



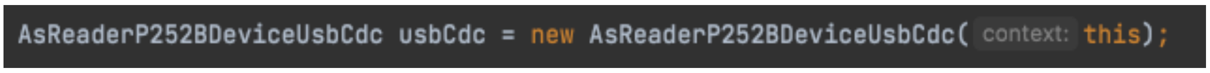
```
@Override
public void onStateChanged(AsReaderP252B reader, AsReaderP252BConnectionState state) {

}

@Override
public void onActionChanged(AsReaderP252B reader, AsReaderP252BActionState action) {

}

@Override
public void onReadTag(AsReaderP252B reader, AsReaderP252BActionState action, String tag, float rssi,

}

@Override
public void onAccessResult(AsReaderP252B reader, AsReaderP252BResultCode code, AsReaderP252BActionSta

}
```

FIG. 1-2-7

1.2.4 Connect to the AsReader Device

1.2.4.1 Connect to the AsReader Device via USB

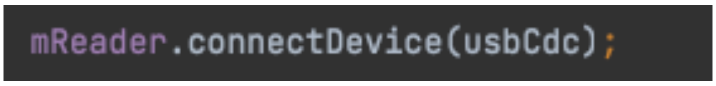
Create the AsReaderP252BDeviceUsbCdc object (FIG. 1-2-8).



```
AsReaderP252BDeviceUsbCdc usbCdc = new AsReaderP252BDeviceUsbCdc( context: this);
```

FIG. 1-2-8

Pass the AsReaderP252BDeviceUsbCdc object as a parameter to the function connectDevice() (FIG. 1-2-9).



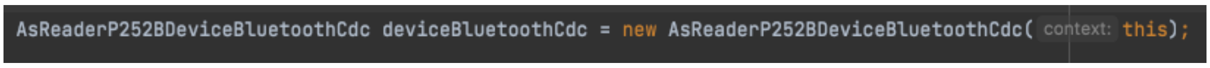
```
mReader.connectDevice(usbCdc);
```

FIG. 1-2-9

The connection status of the AsReader device will be returned via the function onStateChanged (see 2.3.1). If the return value is "Connected", the AsReader device connection is successful.

1.2.4.2 Connect to the AsReader Device via Bluetooth

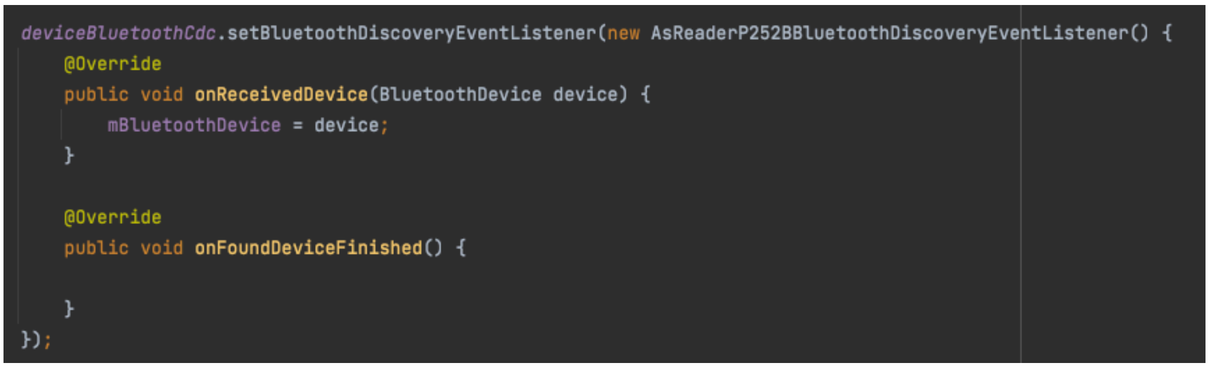
Create the AsReaderP252BDeviceBluetoothCdc object (FIG. 1-2-10).



```
AsReaderP252BDeviceBluetoothCdc deviceBluetoothCdc = new AsReaderP252BDeviceBluetoothCdc( context: this);
```

FIG. 1-2-10

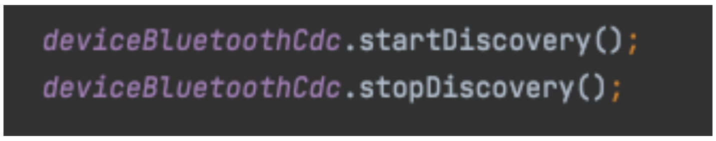
Set function AsReaderP252BBluetoothDiscoveryEventListener (FIG. 1-2-11).



```
deviceBluetoothCdc.setBluetoothDiscoveryEventListener(new AsReaderP252BBluetoothDiscoveryEventListener() {
    @Override
    public void onReceivedDevice(BluetoothDevice device) {
        mBluetoothDevice = device;
    }

    @Override
    public void onFoundDeviceFinished() {

    }
});
```

FIG. 1-2-11

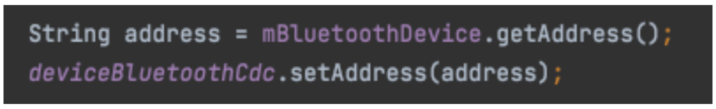
The search for AsReader devices via Bluetooth can be started by calling the function startDiscovery (see 2.6.2). The Bluetooth addresses of AsReader devices can be obtained if the Bluetooth permission is allowed. Stop the search for AsReader devices by calling the function stopDiscovery (see 2.6.3) (FIG.1-2-12).



```
deviceBluetoothCdc.startDiscovery();
deviceBluetoothCdc.stopDiscovery();
```

FIG. 1-2-12

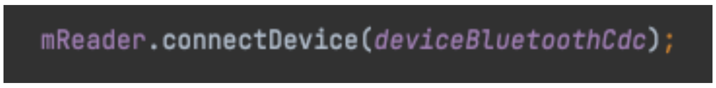
Set the Bluetooth address of the AsReader device as a parameter to the AsReaderP252BDeviceBluetoothCdc object (FIG. 1-2-13).



```
String address = mBluetoothDevice.getAddress();
deviceBluetoothCdc.setAddress(address);
```

FIG. 1-2-13

Pass the AsReaderP252BDeviceBluetoothCdc object as a parameter to the function connectDevice() (FIG. 1-2-14).



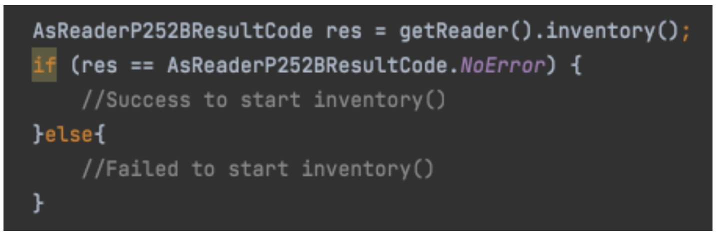
```
mReader.connectDevice(deviceBluetoothCdc);
```

FIG. 1-2-14

The connection status of the AsReader device will be returned via the function onStateChanged (see 2.3.1). If the return value is "Connected", the connection to the AsReader device is successful.

1.2.5 Inventory

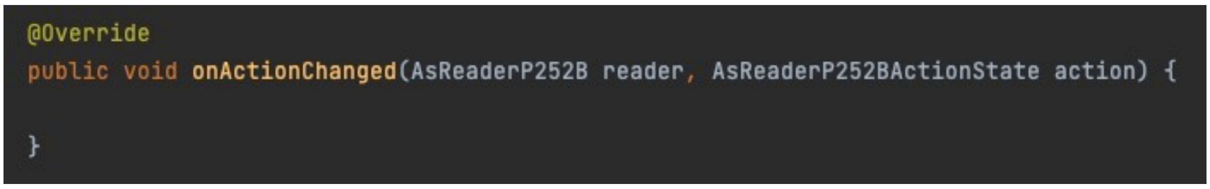
Once the connection is successful, the functions provided in the library can be called. The function “inventory ()” is used as shown in an example below (FIG. 1-2-15).



```
AsReaderP252BResultCode res = getReader().inventory();
if (res == AsReaderP252BResultCode.NoError) {
    //Success to start inventory()
}else{
    //Failed to start inventory()
}
```

FIG. 1-2-15

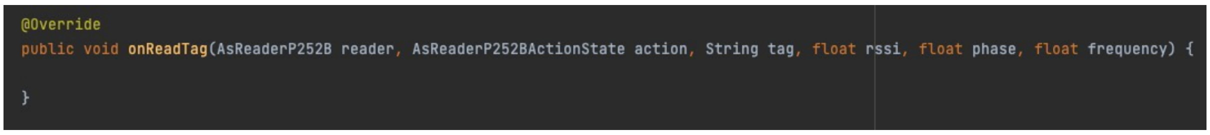
The execution result will be returned via the function onActionChanged (see 2.3.2) (FIG. 1-2-16).



```
@Override
public void onActionChanged(AsReaderP252B reader, AsReaderP252BActionState action) {

}
```

FIG. 1-2-16

The RFID tag data will be returned via the function onReadTag (see 2.3.3) (FIG. 1-2-17).



```
@Override
public void onReadTag(AsReaderP252B reader, AsReaderP252BActionState action, String tag, float rssi, float phase, float frequency) {

}
```

FIG. 1-2-17

2 Functions

2.1 AsReaderP252B

2.1.1 getResultCode

Function	public AsReaderP252BResultCode getResultCode()		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	AsReaderP252BResultCode	Enum AsReaderP252BResultCode (see 2.11.8)

■Function description:
Gets the result of the function execution.

■Sample code:
```
AsReaderP252BResultCode resultcode = asReader.getResultCode();
if (resultcode == AsReaderP252BResultCode.NoError) {
   //Function execution succeeded
} else {
   //Function execution failed
}
```

2.1.2 connectDevice

Function	public void connectDevice(AsReaderP252BDevice asReaderP252BDevice)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
asReaderP252BDevice	IN	AsReaderP252BDevice	The AsReaderP252BDevice object.

■Function description:
Connects to the AsReader device.
Once this function is executed, the connection status will be returned via the function "onStateChanged" (see 2.3.1).

■Sample code:
```
asReader.connectDevice(mDevice);
```

2.1.3 disconnectDevice

Function	public void disconnectDevice()

■Function description:
Disconnects from the AsReader device.
Once this function is executed, the connection status will be returned via the function "onStateChanged" (see 2.3.1).

■Sample code:
```
asReader.disconnectDevice();
```

2.1.4 getCurrentDevice

Function	public AsReaderP252BDevice getCurrentDevice()		
Parameters	IN/OUT	Types	Descriptions
	OUT	AsReaderP252B Device	The AsReaderP252BDevice object. (Note: Returns null if there is no connected device.)

■Function description:
Gets the connected AsReader device.

■Sample code:
AsReaderP252BDevice device = asReader.getCurrentDevice();

2.1.5 getState

Function	public AsReaderP252BConnectionState getState()		
Parameters	IN/OUT	Types	Descriptions
	OUT	AsReaderP252BConnection State	Enum AsReaderP252BConnectionState (see 2.11.1)

■Function description:
Gets the connection status of the AsReader device.

■Sample code:
AsReaderP252BConnectionState connectStage = asReader.getState();

2.1.6 getAction

Function	public AsReaderP252BActionState getAction()		
Parameters	IN/OUT	Types	Descriptions
	OUT	AsReaderP252BAction State	Enum AsReaderP252BActionState (see 2.11.3)

■Function description:
Gets the action status of the AsReader device.

■Sample code:
AsReaderP252BActionState actionStage = asReader.getAction();

2.1.7 getFirmwareVersion

Function	public String getFirmwareVersion()		
Parameters	IN/OUT	Types	Descriptions
	OUT	String	The firmware version of the AsReader device.

■Function description:
Gets the firmware version of the AsReader device.

■Sample code:
String firmwareVersion = asReader.getFirmwareVersion();

2.1.8 getHardwareVersion

Function	public String getHardwareVersion()		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	String	The hardware version of the AsReader device.

■Function description:
Gets the hardware version of the AsReader device.

■Sample code:
```
String hardwareVersion = asReader.getHardwareVersion();
```

2.1.9 getRFModuleVersion

Function	public String getRFModuleVersion()		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	String	The firmware version of the RFID module.

■Function description:
Gets the firmware version of the RFID module.

■Sample code:
```
String rFModuleVersion = asReader.getRFModuleVersion();
```

2.1.10 setEventListener

Function	public void setEventListener(AsReaderP252BEventListener listener)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
listener	IN	AsReaderP252BEvent Listener	The AsReaderP252BEventListener object. (see 2.3)

■Function description:
Sets the AsReaderP252BEventListener object.

■Sample code:
```
asReader.setEventListener(this) ;
```

2.1.11 removeEventListener

Function	public void removeEventListener (AsReaderP252BEventListener listener)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
listener	IN	AsReaderP252BEvent Listener	The AsReaderP252BEventListener object. (see 2.3)

■Function description:
Removes the AsReaderP252BEventListener object.

■Sample code:
```
asReader.removeEventListener(this) ;
```

2.1.12 inventory

Function	public AsReaderP252BResultCode inventory()		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	AsReaderP252B ResultCode	Function execution results. Enum AsReaderP252BResultCode (see 2.11.8)

■Function description:
The AsReader device starts to inventory RFID tags.
Once this function is executed, the "onActionChanged" (see 2.3.2) will be called back, returning the execution result. And the "onReadTag" (see 2.3.3) will be called back, returning the RFID tag data.

■Sample code:

```
AsReaderP252BResultCode resultCode = asReader.inventory();
if (resultcode == AsReaderP252BResultCode.NoError) {
  //Function execution succeeded
} else {
  //Function execution failed
}
```

2.1.13 inventoryHumidityTag

Function	public AsReaderP252BResultCode inventoryHumidityTag()		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	AsReaderP252B ResultCode	Function execution results. Enum AsReaderP252BResultCode (see 2.11.8)

■Function description:
The AsReader device starts to inventory RFID tags with humidity chip.
Once this function is executed, the “onActionChanged” (see 2.3.2) will be called back, returning the execution result. And the “onReadHumidityTag” (see 2.3.6) will be called back, returning the RFID tag data.

■Sample code:
```
AsReaderP252BResultCode resultCode = asReader. inventoryHumidityTag();
if (resultcode == AsReaderP252BResultCode.NoError) {
  //Function execution succeeded
} else {
  //Function execution failed
}
```

2.1.14 inventoryTemperatureTagWithType

Function	public AsReaderP252BResultCode inventoryTemperatureTagWithType(AsReaderP252BTemperatureTagType type)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
type	IN	AsReaderP252BTemperatureTagType	Enum AsReaderP252BTemperatureTagType (see 2.11.10)
	OUT	AsReaderP252BResult Code	Function execution results. Enum AsReaderP252BResultCode (see 2.11.8)

■Function description:
The AsReader device starts to inventory RFID tags with temperature chip.
Once this function is executed, the “onActionChanged” (see 2.3.2) will be called back, returning the execution result. And the “onReadTemperatureTag” (see 2.3.5) will be called back, returning the RFID tag data.

■Sample code:
```
AsReaderP252BResultCode resultCode = asReader.inventoryTemperatureTagWithType
(AsReaderP252BTemperatureTagType.Type_0);
if (resultcode == AsReaderP252BResultCode.NoError) {
  //Function execution succeeded
} else {
  //Function execution failed
}
```

2.1.15 readMemory

Function	public AsReaderP252BResultCode readMemory(AsReaderP252BMemoryBank bank, int offset, int length)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
bank	IN	AsReaderP252BMemoryBank	The memory bank of the RFID tag. Enum AsReaderP252BMemoryBank (see 2.11.5)
offset	IN	int	The start address of the memory bank. (Unit: Word)
length	IN	int	The length of the memory bank to be read. (Unit: Word)
	OUT	AsReaderP252BResultCode	Function execution results. Enum AsReaderP252BResultCode (see 2.11.8)

■Function description:
To read memory bank of the RFID tag.
Once this function is executed, the "onAccessResult" (see 2.3.4) will be called back, returning the execution result.

■Sample code:
```
AsReaderP252BResultCode resultCode = asReader.readMemory
(AsReaderP252BMemoryBank.EPC, 16, 4);
if (resultcode == AsReaderP252BResultCode.NoError) {
  //Function execution succeeded
} else {
  //Function execution failed
}
```

2.1.16 writeMemory

Function	public AsReaderP252BResultCode writeMemory(AsReaderP252BMemoryBank bank, int offset, String data)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
bank	IN	AsReaderP252BMemoryBank	The memory bank of the RFID tag. Enum AsReaderP252BMemoryBank (see 2.11.5)
offset	IN	int	The start address of the memory bank. (Unit: Word)
data	IN	String	The data to be written to the RFID tag. (Hex)
	OUT	AsReaderP252BResultCode	Function execution results. Enum AsReaderP252BResultCode (see 2.11.8)

■Function description:
Writes data to the target memory bank of the RFID tag.
Once this function is executed, the "onAccessResult" (see 2.3.4) will be called back, returning the execution result.

■Sample code:
```
AsReaderP252BResultCode resultCode = asReader.
writeMemory(AsReaderP252BMemoryBank.EPC, 16, "1234");
if (resultcode == AsReaderP252BResultCode.NoError) {
//Function execution succeeded
} else {
//Function execution failed
}
```

2.1.17 lock

Function	public AsReaderP252BResultCode lock(AsReaderP252BLockParam param)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
param	IN	AsReaderP252BLockParam	The AsReaderP252BLockParam object. (see 2.8)
	OUT	AsReaderP252BResultCode	Function execution results. Enum AsReaderP252BResultCode (see 2.11.8)

■Function description:
Locks a target memory bank of the RFID tag.
Once this function is executed, the “onAccessResult” (see 2.3.4) will be called back, returning the execution result.

■Sample code:
```
AsReaderP252BResultCode resultCode = asReader.lock(param);
if (resultcode == AsReaderP252BResultCode.NoError) {
  //Function execution succeeded
} else {
  //Function execution failed
}
```

2.1.18 unlock

Function	public AsReaderP252BResultCode unlock (AsReaderP252BLockParam param)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
param	IN	AsReaderP252BLockParam	The AsReaderP252BLockParam object. (see 2.8)
	OUT	AsReaderP252BResultCode	Function execution results. Enum AsReaderP252BResultCode (see 2.11.8)

■Function description:
Unlocks the locked memory bank of the RFID tag. After unlocking, you can use the default password to overwrite the tag data.
Once this function is executed, the “onAccessResult” (see 2.3.4) will be called back, returning the execution result.

■Sample code:
```
AsReaderP252BResultCode resultCode = asReader.unlock(param);
if (resultcode == AsReaderP252BResultCode.NoError) {
  //Function execution succeeded
} else {
  //Function execution failed
}
```

2.1.19 permaLock

Function	public AsReaderP252BResultCode permaLock(AsReaderP252BLockParam param)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
param	IN	AsReaderP252BLockParam	The AsReaderP252BLockParam object. (see 2.8)
	OUT	AsReaderP252BResultCode	Function execution results. Enum AsReaderP252BResultCode (see 2.11.8)

■Function description:
Permanently locks a target memory bank of the RFID tag.
Permanently locked tag data cannot be changed or unlocked.
Once this function is executed, the “onAccessResult” (see 2.3.4) will be called back, returning the execution result.

■Sample code:

```
AsReaderP252BResultCode resultCode = asReader.permaLock (param);
if (resultcode == AsReaderP252BResultCode.NoError) {
  //Function execution succeeded
} else {
  //Function execution failed
}
```

2.1.20 kill

Function	public AsReaderP252BResultCode kill(String killPassword)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
killPassword	IN	String	Kill password.
	OUT	AsReaderP252BResultCode	Function execution results. Enum AsReaderP252BResultCode (see 2.11.8)

■Function description:
Kills an RFID tag. The killed RFID tag cannot be used.
Once this function is executed, the “onAccessResult” (see 2.3.4) will be called back, returning the execution result.
* The RFID tag cannot be killed when the kill password is "00000000".

■Sample code:

```
AsReaderP252BResultCode resultCode = asReader.kill("12345678");
if (resultcode == AsReaderP252BResultCode.NoError) {
  //Function execution succeeded
} else {
  //Function execution failed
}
```

2.1.21 stop

Function	public AsReaderP252BResultCode stop()		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	AsReaderP252BResultCode	Function execution results. Enum AsReaderP252BResultCode (see 2.11.8)

■Function description:
Stops inventorying RFID tags.
Once this function is executed, the "onActionChanged" (see 2.3.2) will be called back, returning the execution result.

■Sample code:
```
AsReaderP252BResultCode resultCode = asReader.stop();
if (resultcode == AsReaderP252BResultCode.NoError) {
  //Function execution succeeded
} else {
  //Function execution failed
}
```

2.1.22 defaultParameter

Function	public AsReaderP252BResultCode defaultParameter()		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	AsReaderP252BResultCode	Function execution results. Enum AsReaderP252BResultCode (see 2.11.8)

■Function description:
Restores default settings.

■Sample code:
```
AsReaderP252BResultCode resultCode = asReader.defaultParameter();
if (resultcode == AsReaderP252BResultCode.NoError) {
  //Function execution succeeded
} else {
  //Function execution failed
}
```

2.1.23 saveParameter

Function	public AsReaderP252BResultCode saveParameter()		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	AsReaderP252BResultCode	Function execution results. Enum AsReaderP252BResultCode (see 2.11.8)

■Function description:
Saves the parameters to the AsReader device.

■Sample code:
```
AsReaderP252BResultCode resultCode = asReader.saveParameter();
if (resultcode == AsReaderP252BResultCode.NoError) {
  //Function execution succeeded
} else {
  //Function execution failed
}
```

2.1.24 getBuzzer

Function	public AsReaderP252BBuzzerState getBuzzer() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	AsReaderP252BBuzzerState	Enum AsReaderP252BBuzzerState (see 2.11.4)
	-	AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the buzzer status.

■Sample code:
```
try {
        AsReaderP252BBuzzerState mBuzzer = asReader.getBuzzer();
} catch (AsReaderP252BException e) {
}
```

2.1.25 setBuzzer

Function	public void setBuzzer(AsReaderP252BBuzzerState state) throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	IN	AsReaderP252BBuzzerState	Enum AsReaderP252BBuzzerState (see 2.11.4)
	-	AsReaderP252BException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the buzzer status.

■Sample code:
```
try {
        asReader.setBuzzer(AsReaderP252BBuzzerState.Off);
} catch (AsReaderP252BException e) {
}
```

2.1.26 getContinuousMode

Function	public boolean getContinuousMode() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	boolean	true: Continuous mode valid. false: Continuous mode invalid.
	-	AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets whether to inventory continuously.

■Sample code:
```
try {
        boolean continuousMode = asReader.getContinuousMode();
} catch (AsReaderP252BException e) {
}
```

2.1.27 setContinuousMode

Function	public void setContinuousMode(boolean enabled) throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
enable	IN	boolean	true: Continuous mode valid. false: Continuous mode invalid.
	-	AsReaderP252BException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets whether to inventory continuously.

■Sample code:
```
try {
	asReader.setContinuousMode(true);
} catch (AsReaderP252BException e) {
}
```

2.1.28 getPowerGain

Function	public int getPowerGain() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	int	The value of power. (Unit: dBm × 10)
	-	AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the power of the AsReader device.

■Sample code:
```
try {
	int powerGain = asReader.getPowerGain();
} catch (AsReaderP252BException e) {
}
```

2.1.29 setPowerGain

Function	public void setPowerGain(int power) throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
power	IN	int	The value of power. (Unit: dBm × 10) Range: 2~30.
	-	AsReaderP252BException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the power of the AsReader device.

■Sample code:
```
try {
	asReader.setPowerGain(powerGain);
} catch (AsReaderP252BException e) {
}
```

2.1.30 getPowerGainRange

Function	public AsReaderP252BPowerRange getPowerGainRange()		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	AsReaderP252BPowerRange	The power range can be set. The AsReaderP252BPowerRange object. (see 2.9)

■Function description:
Gets the power range of the AsReader device.

■Sample code:
```
try {
      AsReaderP252BPowerRange powerGainRange = asReader.getPowerGainRange();
      int min = powerGainRange.min; // The minimum value of the power.
      int max = powerGainRange.max; // The maximum value of the power.
} catch (AsReaderP252BException e) {
}
```

2.1.31 getOperationTime

Function	public int getOperationTime() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	int	Duration of inventory. (Unit: ms)
	-	AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the duration of an inventory.

■Sample code:
```
try {
     int mOperationTime = asReader.getOperationTime();
} catch (AsReaderP252BException e) {
}
```

2.1.32 setOperationTime

Function	public void setOperationTime(int time) throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
time	IN	int	Duration of inventory. (Unit: ms)
	-	AsReaderP252BException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the duration of an inventory.

■Sample code:
```
try {
     asReader.setOperationTime(200);
} catch (AsReaderP252BException e) {
}
```

2.1.33 getIdleTime

Function	public int getIdleTime() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	int	The duration of the radio waves non-emitted. (Unit: 10ms)
	-	AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the duration of the radio waves non-emitted when inventorying.

■Sample code:
```
try {
    int idleTime = asReader.getIdleTime();
} catch (AsReaderP252BException e) {
}
```

2.1.34 setIdleTime

Function	public void setIdleTime(int time) throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
time	IN	int	The duration of the radio waves non-emitted. (Unit: 10ms). Range: 0~65535.
	-	AsReaderP252BException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the duration of the radio waves non-emitted when inventorying.

■Sample code:
```
try {
    asReader.setIdleTime(300);
} catch (AsReaderP252BException e) {i
}
```

2.1.35 getAutoOffTime

Function	public int getAutoOffTime () throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	int	Auto off time. (Unit: s)
	-	AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the auto off time when the AsReader device is not connected.

■Sample code:
```
try {
    int autoOffTime = asReader.getAutoOffTime();
} catch (AsReaderP252BException e) {
}
```

2.1.36 setAutoOffTime

Function	public void setAutoOffTime(int time) throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
time	IN	int	Auto off time. (Unit: s) Range: 0~1800.
	-	AsReaderP252BException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the auto off time when the AsReader device is not connected.

■Sample code:

```
try {
    asReader.setAutoOffTime(200);
} catch (AsReaderP252BException e) {
}
```

2.1.37 getSleepTime

Function	public int getSleepTime() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	int	Auto sleep time. (Unit: s)
	-	AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the auto sleep time when the AsReader device is not connected.

■Sample code:

```
try {
    int mSleepTime = asReader.getSleepTime();
} catch (AsReaderP252BException e) {
}
```

2.1.38 setSleepTime

Function	public void setSleepTime(int time) throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
time	IN	int	Auto sleep time. (Unit: s) Range: 0~1800.
	-	AsReaderP252BException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the auto sleep time when the AsReader device is not connected.

■Sample code:

```
try {
    asReader.setSleepTime(200);
} catch (AsReaderP252BException e) {
}
```

2.1.39 getBarcodeTimeOut

Function	public int getBarcodeTimeOut() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	int	The timeout when the AsReader device scans a barcode. (Unit: s)
	-	AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the timeout when the AsReader device scans a barcode.

■Sample code:
```
try {
     int mBarcodeTimeOut= asReader.getBarcodeTimeOut();
} catch (AsReaderP252BException e) {
}
```

2.1.40 setBarcodeTimeOut

Function	public void setBarcodeTimeOut(int time) throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
time	IN	int	The timeout when the AsReader device scans a barcode. (Unit: s) Range: 4~300.
	-	AsReaderP252BException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the timeout when the AsReader device scans a barcode.

■Sample code:
```
try {
     asReader.setBarcodeTimeOut(10);
} catch (AsReaderP252BException e) {
}
```

2.1.41 getBaudRateList

Function	public ArrayList<String> getBaudRateList() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	ArrayList	The baud rate list.
	-	AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the baud rate list of the AsReader device.

■Sample code:
```
try {
      ArrayList mBaudRateList = asReader.getBaudRateList();
} catch (AsReaderP252BException e) {
}
```

2.1.42 getBaudRate

Function	public int getBaudRate() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	int	The index of baud rate list.
	-	AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the index of baud rate list.

■Sample code:
```
try {
     int mBaudRate = asReader.getBaudRate();
} catch (AsReaderP252BException e) {
}
```

2.1.43 setBaudRate

Function	public void setBaudRate(int rate) throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
rate	IN	int	The index of baud rate list.
	-	AsReaderP252BException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the index of baud rate list.

■Sample code:
```
try {
     asReader.setBaudRate(1);
} catch (AsReaderP252BException e) {
}
```

2.1.44 getAccessPassword

Function	public String getAccessPassword() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	String	Access password.
	-	AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the access password required for the AsReader device to operate on a locked RFID tag.

■Sample code:
```
try {
     String mAccessPassword = asReader.getAccessPassword();
} catch (AsReaderP252BException e) {
}
```

2.1.45 setAccessPassword

Function	public void setAccessPassword(String password) throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
password	IN	String	Access password.
	-	AsReaderP252BException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the access password required for the AsReader device to operate on a locked RFID tag.
When the access password is "00000000", the password is the initial state.

■Sample code:
```
try {
    asReader.setAccessPassword("12345678");
} catch (AsReaderP252BException e) {
}
```

2.1.46 getQuerySession

Function	public AsReaderP252BQuerySession getQuerySession() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	AsReaderP252BQuerySession	Enum AsReaderP252BQuerySession (see 2.11.6)
	-	AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the session value of the inventory.

■Sample code:
```
try {
    AsReaderP252BQuerySession mSession = asReader.getQuerySession();
} catch (AsReaderP252BException e) {
}
```

2.1.47 setQuerySession

Function	public void setQuerySession(AsReaderP252BQuerySession session) throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
session	IN	AsReaderP252BQuerySession	Enum AsReaderP252BQuerySession (see 2.11.6)
	-	AsReaderP252BException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the session value of the inventory.

■Sample code:
```
try {
    asReader.setQuerySession(AsReaderP252BQuerySession.S0);
} catch (AsReaderP252BException e) {
}
```

2.1.48 getSessionFlag

Function	public AsReaderP252BSessionFlag getSessionFlag() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	AsReaderP252BSessionFlag	Enum AsReaderP252BSessionFlag (see 2.11.7)
	-	AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the session flag of the inventory.

■Sample code:
```
try {
    AsReaderP252BSessionFlag mTarget = asReader.getSessionFlag();
} catch (AsReaderP252BException e) {
}
```

2.1.49 setSessionFlag

Function	public void setSessionFlag(AsReaderP252BSessionFlag target) throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
target	IN	AsReaderP252BSessionFlag	Enum AsReaderP252BSessionFlag (see 2.11.7)
	-	AsReaderP252BException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the session flag of the inventory.

■Sample code:
```
try {
    asReader.setSessionFlag(AsReaderP252BSessionFlag.AB);
} catch (AsReaderP252BException e) {
}
```

2.1.50 getLinkProfile

Function	public int getLinkProfile() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	int	Link profile value.
	-	AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the link profile value of the AsReader device.

■Sample code:
```
try {
    int mLinkProfile = asReader.getLinkProfile();
} catch (AsReaderP252BException e) {
}
```

2.1.51 setLinkProfile

Function	public void setLinkProfile(int value) throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
value	IN	int	Link profile value. Range: 0~3.
	-	AsReaderP252BException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the link profile value of the AsReader device.

■Sample code:
```
try {
     asReader.setLinkProfile(1);
} catch (AsReaderP252BException e) {
}
```

2.1.52 getQValue

Function	public int getQValue() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	int	The Q value of the inventory.
	-	AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the Q value of the inventory.

■Sample code:
```
try {
     int mQValue = asReader.getQValue();
} catch (AsReaderP252BException e) {
}
```

2.1.53 setQValue

Function	public void setQValue(int value) throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
value	IN	int	The Q value of the inventory. Range: 0~15.
	-	AsReaderP252BException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the Q value of the inventory.

■Sample code:
```
try {
     asReader.setQValue(10);
} catch (AsReaderP252BException e) {
}
```

2.1.54 getMaxQ

Function	public int getMaxQ() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	int	The maximum Q value of the inventory.
	-	AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the maximum Q value of the inventory.

■Sample code:
```
try {
    int maxQ = asReader.getMaxQ();
} catch (AsReaderP252BException e) {
}
```

2.1.55 getMinQ

Function	public int getMinQ() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	int	The minimum Q value of the inventory.
	-	AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the minimum Q value of the inventory.

■Sample code:
```
try {
    int minQ = asReader.getMinQ();
} catch (AsReaderP252BException e) {
}
```

2.1.56 getSerialNumber

Function	public String getSerialNumber() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	String	The serial number of the AsReader device.
	-	AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the serial number of the AsReader device.

■Sample code:
```
try {
    String mSerialNumber = asReader.getSerialNumber();
} catch (AsReaderP252BException e) {
}
```

2.1.57 getBatteryStatus

Function	public int getBatteryStatus() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	int	The battery power level of the AsReader device. Range: 0, 1, 2, 3, 4
	-	AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the battery power level of the AsReader device.
0: 0
1: 25%
2: 50%
3: 75%
4: 100%

■Sample code:
```
try {
    int mBattery = asReader.getBatteryStatus();
} catch (AsReaderP252BException e) {
}
```

2.1.58 getReportRSSI

Function	public boolean getReportRSSI() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	boolean	true: Get RSSI data. false: Do not get RSSI data.
	-	AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets whether to get RSSI data when inventorying RFID tags.

■Sample code:
```
try {
    boolean mReportRSSI = asReader.getReportRSSI();
} catch (AsReaderP252BException e) {
}
```

2.1.59 setReportRSSI

Function	public void setReportRSSI(boolean enabled) throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
enabled	IN	boolean	true: Get RSSI data. false: Do not get RSSI data.
	-	AsReaderP252BException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets whether to get RSSI data when inventorying RFID tags.

■Sample code:
```
try {
    asReader.setReportRSSI(true);
} catch (AsReaderP252BException e) {
}
```

2.1.60 clearEpcMask

Function	public void clearEpcMask() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	-	AsReaderP252BException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Removes the EPC mask data.

■Sample code:
```
try {
    asReader.clearEpcMask();
} catch (AsReaderP252BException e) {
}
```

2.1.61 getEpcMaskCount

Function	public int getEpcMaskCount() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	int	Number of EPC masks.
		AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the number of EPC masks.

■Sample code:
```
try {
    int mCount = asReader.getEpcMaskCount();
} catch (AsReaderP252BException e) {
}
```

2.1.62 addEpcMask

Function	public void addEpcMask(AsReaderP252BSelectMaskEpcParam param) throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
param	IN	AsReaderP252BSelectMaskEpcParam	The AsReaderP252BSelectMaskEpcParam object. (see 2.10)
		AsReaderP252BException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Adds an EPC mask.

■Sample code:
```
try {
    asReader.addEpcMask(mask);
} catch (AsReaderP252BException e) {
}
```

2.1.63 getEpcMask

Function	public AsReaderP252BSelectMaskEpcParam getEpcMask(int index) throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
index	IN	int	Index of the EPC mask.
	OUT	AsReaderP252BSelectMaskEpcParam	Function execution results. The AsReaderP252BSelectMaskEpcParam object. (see 2.10)
		AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the EPC mask for the specified index.

■Sample code:
```
try {
    AsReaderP252BSelectMaskEpcParam mask = asReader. getEpcMask(0);
} catch (AsReaderP252BException e) {
}
```

2.1.64 getFrequencyAutomatic

Function	public boolean getFrequencyAutomatic() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	Boolean	Whether to use frequency hopping. YES: Use. NO: Do not use.
		AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets whether to use frequency hopping when inventorying.

■Sample code:
```
try {
    boolean mFrequencyAutomatic = asReader.getFrequencyAutomatic();
} catch (AsReaderP252BException e) {
}
```

2.1.65 setFrequencyAutomatic

Function	public void setFrequencyAutomatic(boolean isAutomatic) throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
isAutomatic	IN	boolean	Whether to use frequency hopping. YES: Use. NO: Do not use.
		AsReaderP252BException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets whether to use frequency hopping when inventorying.

■Sample code:
```
try {
    asReader.setFrequencyAutomatic(true);
} catch (AsReaderP252BException e) {
}
```

2.1.66 getFrequencyList

Function	public AsReaderP252BLbtItem[]getFrequencyList() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	AsReaderP252BLbtItem[]	The frequency list.
		AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the frequency list of the AsReader device.

■Sample code:
```
try {
    AsReaderP252BLbtItem[] mItems = asReader. getFrequencyList();
} catch (AsReaderP252BException e) {
}
```

2.1.67 setFrequencyList

Function	public void setFrequencyList(AsReaderP252BLbtItem[] table) throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
table	IN	AsReaderP252BLbtItem[]	The frequency list.
		AsReaderP252BException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the frequency list of the AsReader device.

■Sample code:
```
try {
    asReader. setFrequencyList(table);
} catch (AsReaderP252BException e) {
}
```

2.1.68 startDecode

Function	public AsReaderP252BResultCode startDecode();		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	AsReaderP252BResultCode	Function execution results. Enum AsReaderP252BResultCode (see 2.11.8)

■Function description:
The AsReader device starts to scan barcodes.
Once this function is executed, the “onReadBarcode” (see 2.3.9) will be called back, returning the barcode data.

■Sample code:
```
AsReaderP252BResultCode resultCode = asReader.startDecode();
if (resultCode == AsReaderP252BResultCode.NoError) {
    //Function execution succeeded
}else{
    //Function execution failed
}
```

2.1.69 stopDecode

Function	public AsReaderP252BResultCode stopDecode()		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	AsReaderP252BResultCode	Function execution results. Enum AsReaderP252BResultCode (see 2.11.8)

■Function description:
Stop scanning barcodes.

■Sample code:

```
AsReaderP252BResultCode resultCode = asReader.stopDecode();
if (resultCode == AsReaderP252BResultCode.NoError) {
   //Function execution succeeded
}else{
   //Function execution failed
}
```

2.1.70 setCharset

Function	public void setCharset(Charset charset) throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
charset	IN	Charset	Charset.
		AsReaderP252BException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets the encoding type of barcode scanning.

■Sample code:

```
try {
       asReader.setCharset(Charset.forName("ASCII"));
} catch (AsReaderP252BException e) {
}
```

2.1.71 getRegion

Function	public String getRegion() throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	String	The region or country where the AsReader device is set to meet local regulations.
		AsReaderP252BException	If a fault occurs while getting the parameter, an exception is returned.

■Function description:
Gets the region or country where the AsReader device is set to meet local regulations.

■Sample code:

```
try {
  String mGlobalBand = asReader.getRegion();
} catch (AsReaderP252BException e) {
}
```

2.1.72 setScanMode

Function	public void setScanMode(AsReaderP252BScanMode scanMode)throws AsReaderP252BException		
Parameters	**IN/OUT**	**Types**	**Descriptions**
scanMode	IN	AsReaderP252BScanMode	Barcode mode or RFID mode. Enum AsReaderP252BScanMode (see 2.11.9)
		AsReaderP252BException	If a fault occurs while setting the parameter, an exception is returned.

■Function description:
Sets whether the AsReader device is in barcode mode or RFID mode.

■Sample code:

```
try {
   asReader.setScanMode(AsReaderP252BScanMode.RFIDScanMode);
} catch (AsReaderP252BException e) {
}
```

2.2 AsReaderP252BManager

2.2.1 getInstance

Function	public static AsReaderP252B getInstance()		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	AsReaderP252B	The AsReaderP252B object. (see 2.1)

■Function description:
Gets the AsReaderP252B objects.

■Sample code:
AsReaderP252B asReader = AsReaderP252BManager.getInstance();

2.2.2 onDestroy

Function	public static void onDestroy()

■Function description:
Destroy the AsReaderP252B objects and release resources.

■Sample code:
AsReaderP252BManager.onDestroy();

2.2.3 getVersion

Function	public static String getVersion()		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	String	The SDK version.

■Function description:
Gets the SDK version.

■Sample code:
String version = AsReaderP252BManager.getVersion();

2.3 AsReaderP252BEventListener

2.3.1 onStateChanged

Function	void onStateChanged(AsReaderP252B reader, AsReaderP252BConnectionState state)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
reader	OUT	AsReaderP252B	The AsReaderP252B object.
state	OUT	AsReaderP252BConnectionState	Connection status. Enum AsReaderP252BConnectionState (see 2.11.1)

■Function description:
The function will be called back when the connection status of the AsReader device changes. Once the function "connectDevice" (see 2.1.2), "disconnectDevice" (see 2.1.3) is executed, the connection status will be returned via this function.

■Sample code:
```
@Override
public void onStateChanged(AsReaderP252B reader, AsReaderP252BConnectionState state) {
//reader: The AsReaderP252B object
//state: The connection status of the AsReader device
}
```

2.3.2 onActionChanged

Function	void onActionChanged(AsReaderP252B reader, AsReaderP252BActionState action)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
reader	OUT	AsReaderP252B	The AsReaderP252B object.
action	OUT	AsReaderP252BActionState	Enum AsReaderP252BActionState (see 2.11.3)

■Function description:
The function will be called back when the action status of the AsReader device changes. Once the function "inventory" (see 2.1.12), inventoryHumidityTag (see 2.1.13), inventoryTemperatureTagWithType (see 2.1.14), or "stop" (see 2.1.21) is executed, the execution result will be returned via this function.

■Sample code:
```
@Override
public onActionChanged(AsReaderP252B reader, AsReaderP252BActionState action) {
	//reader: The AsReaderP252B object
	//action: The action status of AsReader device
}
```

2.3.3 onReadTag

Function	void onReadTag(AsReaderP252B reader, AsReaderP252BActionState action, String tag, float rssi, float phase, float frequency)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
reader	OUT	AsReaderP252B	The AsReaderP252B object.
action	OUT	AsReaderP252BActionState	Enum AsReaderP252BActionState (see 2.11.3)
tag	OUT	String	The PCEPC data of RFID tag. (Hex)
rssi	OUT	float	The RSSI data of RFID tag.
phase	OUT	float	The phase value of RFID tag.
frequency	OUT	float	The frequency of RFID tag when it is inventoried.

■Function description:
The function will be called back when the RFID tags are read.
Once the function “inventory” (see 2.1.12) is executed or the Trigger button is pressed, the RFID tag data will be returned via this function.

■Sample code:

```
@Override
public void onReadTag(AsReaderP252B reader, AsReaderP252BActionState action, String tag,
float rssi, float phase, float frequency) {
   //reader: The AsReaderP252B object
   //action: The action status of the AsReader device
   //tag: The PCEPC data of RFID tag (Hex)
   //rssi: The RSSI data of RFID tag
   //phase: The phase value of RFID tag
   //frequency: The frequency of RFID tag when it is inventoried
}
```

2.3.4 onAccessResult

Function	void onAccessResult(AsReaderP252B reader, AsReaderP252BResultCode code, AsReaderP252BActionState action, String epc, String data, float rssi, float phase, float frequency)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
reader	OUT	AsReaderP252B	The AsReaderP252B object.
code	OUT	AsReaderP252BResultCode	Function execution results. Enum AsReaderP252BResultCode (see 2.11.8)
action	OUT	AsReaderP252BActionState	Enum AsReaderP252BActionState (see 2.11.3)
epc	OUT	String	The EPC data of RFID tag. (Hex)
data	OUT	String	The read RFID tag data. (Hex)
rssi	OUT	float	The RSSI data of RFID tag.
phase	OUT	float	The phase value of RFID tag.
frequency	OUT	float	The frequency of RFID tag when it is inventoried.

■Function description:
The function will be called back when function “readMemory” (see 2.1.15), “writeMemory” (see 2.1.16), “lock” (see 2.1.17), “unlock” (see 2.1.18), “permaLock” (see 2.1.19), or “kill” (see 2.1.20) is executed, returning the execution result.

■Sample code:

```
@Override
public void onAccessResult(AsReaderP252B reader, AsReaderP252BResultCode code,
AsReaderP252BActionState action, String epc, String data, float rssi, float phase, float frequency) {
        //reader: The AsReaderP252B object
        //code: Function execution result
        //action: The action status of the AsReader device
        //epc: The EPC data of RFID tag (Hex)
        //data: The read RFID tag data (Hex)
        //rssi: The RSSI data of RFID tag
        //phase: The phase value of RFID tag
        //frequency: The frequency of RFID tag when it is inventoried
}
```

2.3.5 onReadTemperatureTag

Function	void onReadTemperatureTag(AsReaderP252B reader, AsReaderP252BActionState action, String tag, float rssi, float phase, float frequency, float temperature)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
reader	OUT	AsReaderP252B	The AsReaderP252B object.
action	OUT	AsReaderP252BActionState	Enum AsReaderP252BActionState (see 2.11.3)
tag	OUT	String	The PCEPC data of RFID tag. (Hex)
rssi	OUT	float	The RSSI data of RFID tag.
phase	OUT	float	The phase value of RFID tag.
frequency	OUT	float	The frequency of RFID tag when it is inventoried.
temperature	OUT	float	The temperature.

■Function description:
Receives the RFID tag data with temperature chip.
Once the function "inventoryTemperatureTagWithType" (see 2.1.14) is executed or the Trigger button is pressed, the RFID tag data will be returned via this function.

■Sample code:

```
@Override
public onReadTemperatureTag(AsReaderP252B reader, AsReaderP252BActionState action, String
tag, float rssi, float phase, float frequency, float temperature) {
   //reader: The AsReaderP252B object
   //action: The action status of the AsReader device
   //tag: The PCEPC data of RFID tag (Hex)
   //rssi: The RSSI data of RFID tag
   //phase: The phase value of RFID tag
   //frequency: The frequency of RFID tag when it is inventoried.
   //temperature: The temperature
}
```

2.3.6 onReadHumidityTag

Function	void onReadHumidityTag(AsReaderP252B reader, AsReaderP252BActionState action, String tag, float rssi, float phase, float frequency, float humidity)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
reader	OUT	AsReaderP252B	The AsReaderP252B object.
action	OUT	AsReaderP252BActionState	Enum AsReaderP252BActionState (see 2.11.3)
tag	OUT	String	The PCEPC data of RFID tag. (Hex)
rssi	OUT	float	The RSSI data of RFID tag.
phase	OUT	float	The phase value of RFID tag.
frequency	OUT	float	The frequency of RFID tag when it is inventoried.
humidity	OUT	float	The humidity.

■Function description:
Receives the RFID tag data with humidity chip.
Once the function “inventoryHumidityTag” (see 2.1.13) is executed or the Trigger button is pressed, the RFID tag data will be returned via this function.

■Sample code:
```
@Override
public onReadHumidityTag(AsReaderP252B reader, AsReaderP252BActionState action, String tag,
float rssi, float phase, float frequency, float humidity) {
   //reader: The AsReaderP252B object
   //action: The action status of the AsReader device
   //tag: The PCEPC data of RFID tag. (Hex)
   //rssi: The RSSI data of RFID tag
   //phase: The phase value of RFID tag
   //frequency: The frequency of RFID tag when it is inventoried
   //humidity: The humidity
}
```

2.3.7 onKeyEvent

Function	boolean onKeyEvent(AsReaderP252B reader, AsReaderP252BKeyState state)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
reader	OUT	AsReader	The AsReaderP252B object.
state	OUT	AsReaderP252BKeyState	The status of the Trigger button. Enum AsReaderP252BKeyState (see 2.11.2)
	IN	boolean	YES: Execute the default actions of SDK. Press the Trigger button: Start inventory/start scanning. Release the Trigger button: Stop inventory/stop scanning. NO: Default actions of SDK are not executed.

■Function description:
Once the Trigger button of the AsReader device is pressed or released, this function will be called back.

■Sample code:
```
@Override
public boolean onKeyEvent(AsReaderP252B reader, AsReaderP252BKeyState state) {
       //reader: The AsReaderP252B object
       //state: The status of the Trigger button
}
```

2.3.8 onModeKeyEvent

Function	boolean onModeKeyEvent(AsReaderP252B reader, AsReaderP252BKeyState state)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
reader	OUT	AsReaderP252B	The AsReaderP252B object.
state	OUT	AsReaderP252BKeyState	The status of the Mode button. Enum AsReaderP252BKeyState (see 2.11.2)
	IN	boolean	YES: Switch the Barcode mode/RFID mode. NO: Keep the current mode.

■Function description:
Once the Mode button of the AsReader device is pressed or released, this function will be called back.

■Sample code:
```
@Override
public boolean onModeKeyEvent(AsReaderP252B reader, AsReaderP252BKeyState state){
        //reader: The AsReaderP252B object
        //state: The status of the Mode button.
}
```

2.3.9 onReadBarcode

Function	void onReadBarcode(AsReaderP252B reader, AsReaderP252BBarcodeType type, byte[] barcodeData)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
reader	OUT	AsReaderP252B	The AsReaderP252B object.
type	OUT	AsReaderP252BBarcodeType	The barcode type. Enum AsReaderP252BBarcodeType (see 2.11.11)
barcodeData	OUT	byte[]	The scanned barcode data.

■Function description:
Receives scanned barcode data.
Once the function "startDecode" (see 2.1.68) is executed or the Trigger button is pressed, the barcode data will be returned via this function.

■Sample code:
```
@Override
void onReadBarcode(AsReaderP252B reader, AsReaderP252BBarcodeType type, byte[]
barcodeData) {
      //reader: The AsReaderP252B object
      // type: The barcode type
      // barcodeData: The scanned barcode data
      //String string = new String(barcodeData, Charset.forName("ASCII"));
}
```

2.3.10 onReceivedData

Function	void onReceivedData(byte[] data)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
data	OUT	byte[]	Command returned by the AsReader device.

■Function description:
Receives the command returned by the AsReader device.

■Sample code:
```
@Override
void onReceivedData(byte[] data){
      // data: Command returned by the AsReader device
}
```

2.4 AsReaderP252BBluetoothDiscoveryEventListener

2.4.1 onReceivedDevice

Function	void onReceivedDevice(BluetoothDevice device);		
Parameters	**IN/OUT**	**Types**	**Descriptions**
device	OUT	BluetoothDevice	The AsReader device found via Bluetooth.

■Function description:
Receives the AsReader device found via Bluetooth.
Once the function "startDiscovery" (see 2.6.2) is executed, the AsReader device found via Bluetooth will be returned via this function.

■Sample code:
```
@Override
public void onReceivedDevice(final BluetoothDevice device) {
  //device: The AsReader device found via Bluetooth
}
```

2.4.2 onFoundDeviceFinished

Function	void onFoundDeviceFinished();

■Function description:
Complete the search for AsReader device via Bluetooth.
Once the function "stopDiscovery" (see 2.6.3) is executed or no AsReader device found within 15 seconds, the status of stopping the search will be returned via this function.

■Sample code:
```
@Override
public void onFoundDeviceFinished() {
}
```

2.5 AsReaderP252BDeviceUsbCdc

2.5.1 AsReaderP252BDeviceUsbCdc

Function	public AsReaderP252BDeviceUsbCdc(Context context)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
context	IN	Context	The AsReaderP252BDeviceUsbCdc object.

■Function description:
Creates the AsReaderP252BDeviceUsbCdc object.

■Sample code:
```
AsReaderP252BDeviceUsbCdc usbCdc = new AsReaderP252BDeviceUsbCdc(this);
```

2.6 AsReaderP252BDeviceBluetoothCdc

2.6.1 AsReaderP252BDeviceBluetoothCdc

Function	public AsReaderP252BDeviceBluetoothCdc(Context context)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
context	IN	Context	The AsReaderP252BDeviceBluetoothCdc object.

■Function description:
Creates the AsReaderP252BDeviceBluetoothCdc object.

■Sample code:
```
AsReaderP252BDeviceBluetoothCdc bluetoothCdc = new
AsReaderP252BDeviceBluetoothCdc(this);
```

2.6.2 startDiscovery

Function	public void startDiscovery()

■Function description:
Starts to search the AsReader device via Bluetooth.
Once this function is executed, the “onReceivedDevice” (see 2.4.1) will be called back, returning the AsReader device searched via Bluetooth.

■Sample code:
```
bluetoothCdc.startDiscovery();
```

2.6.3 stopDiscovery

Function	public void stopDiscovery()

■Function description:
Stops to search the AsReader device via Bluetooth.
Once this function is executed, the “onFoundDeviceFinished” (see 2.4.2) will be called back, returning the status of stopping the search.

■Sample code:
```
bluetoothCdc.stopDiscovery();
```

2.7 AsReaderP252BLbtItem

2.7.1 getSlot

Function	public int getSlot()		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	int	The frequency position of the LBT frequency list.

■Function description:
Gets the frequency position of the LBT frequency list.

■Sample code:
int mSlot = item.getSlot();

2.7.2 isUsed

Function	public boolean isUsed()		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	boolean	true: Valid. false: Invalid.

■Function description:
Gets whether the LBT is valid.

■Sample code:
boolean mIsUsed = item.isUsed();

2.7.3 setUsed

Function	public void setUsed(boolean used)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
used	IN	boolean	true: Valid. false: Invalid.

■Function description:
Sets whether the LBT is valid.

■Sample code:
item.setUsed(true);

2.7.4 getFrequency

Function	public String getFrequency()		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	String	The frequency of the LBT frequency list.

■Function description:
Gets the frequency of the LBT frequency list.

■Sample code:
String mFrequency= item.getFrequency();

2.8 AsReaderP252BLockParam

2.8.1 AsReaderP252BLockParam

Function	public AsReaderP252BLockParam(boolean killPassword, boolean accessPassword, boolean epc, boolean tid, boolean user)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
killPassword	IN	boolean	true: Lock the kill password. false: Do not lock the kill password.
accessPassword	IN	boolean	true: Lock the access password. false: Do not lock the access password.
epc	IN	boolean	true: Lock the EPC bank. false: Do not lock the EPC bank.
tid	IN	boolean	true: Lock the TID bank. false: Do not lock the TID bank.
user	IN	boolean	true: Lock the User bank. false: Do not lock the User bank.
	OUT	AsReaderP252BLockParam	The AsReaderP252BLockParam object.

■Function description:
Creates the AsReaderP252BLockParam object.

■Sample code:
```
AsReaderP252BLockParam param = new AsReaderP252BLockParam(true, true, true, true, true);
```

2.9 AsReaderP252BPowerRange

2.9.1 getMin

Function	public int getMin()		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	int	The minimum value of the power.

■Function description:
Gets the minimum value of the power.

■Sample code:
```
int min = powerGainRange.min;
```

2.9.2 getMax

Function	public int getMax()		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	int	The maximum value of the power.

■Function description:
Gets the maximum value of the power.

■Sample code:
```
int max = powerGainRange.max;
```

2.10 AsReaderP252BSelectMaskEpcParam

2.10.1 AsReaderP252BSelectMaskEpcParam

Function	public AsReaderP252BSelectMaskEpcParam()		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	AsReaderP252BSelectMask EpcParam	The AsReaderP252BSelectMaskEpcParam object.

■Function description:
Creates the AsReaderP252BSelectMaskEpcParam object.

■Sample code:
AsReaderP252BSelectMaskEpcParam selectMaskEpcParam = new AsReaderP252BSelectMaskEpcParam();

2.10.2 getOffset

Function	public int getOffset()		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	int	The start address of the mask.

■Function description:
Gets the start address of the mask.

■Sample code:
int mOffset = selectMaskEpcParam.getOffset();

2.10.3 setOffset

Function	public void setOffset(int offset)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
offset	IN	int	The start address of the mask.

■Function description:
Sets the start address of the mask.

■Sample code:
selectMaskEpcParam.setOffset(1);

2.10.4 getLength

Function	public int getLength()		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	int	The length of the mask.

■Function description:
Gets the length of the mask.

■Sample code:
int mLength = selectMaskEpcParam.getLength();

2.10.5 setLength

Function	public void setLength(int length)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
length	IN	int	The length of the mask.

■Function description:
Sets the length of the mask.

■Sample code:
```
selectMaskEpcParam.setLength(1);
```

2.10.6 getMask

Function	public String getMask()		
Parameters	**IN/OUT**	**Types**	**Descriptions**
	OUT	String	The mask data. (Hex)

■Function description:
Gets the mask data.

■Sample code:
```
String mMask = selectMaskEpcParam.getMask();
```

2.10.7 setMask

Function	public void setMask(String mask)		
Parameters	**IN/OUT**	**Types**	**Descriptions**
mask	IN	String	The mask data. (Hex)

■Function description:
Sets the mask data.

■Sample code:
```
selectMaskEpcParam.setMask(“1111”);
```

2.11 Enum

2.11.1 AsReaderP252BConnectionState

Definitions	Descriptions
Disconnected = 0	Disconnected.
Connecting= 2	Connecting.
Connected = 3	Connected.
Cancelling = 4	Cancel the connection.

2.11.2 AsReaderP252BKeyState

Definitions	Descriptions
KeyUp = 0	The button is released.
KeyDown = 1	The button is pressed.

2.11.3 AsReaderP252BActionState

Definitions	Descriptions
Stop = 0x73	Operation Stopped.
Inventory = 0x66	Inventory in progress.
ReadMemory = 0x72	Read Memory in progress.
WriteMemory = 0x77	Write Memory in progress
Lock = 0x6C	Lock in progress.
Unlock = 0x6D	Unlock in progress.
PermaLock = 0x70	Perma Lock in progress.
Kill = 0x6B	Kill Tag in progress.
StartDecode = 0x64	Scan Barcode in progress.
StartBuzzer = 0x75	Start Buzzer in progress.
StartVibrator = 0x76	Start Vibrator in progress.
WaitForResponse = 0xF0	Wait for response.

2.11.4 AsReaderP252BBuzzerState

Definitions	Descriptions
Off = 0	Turn off the buzzer.
Low = 1	Buzzer low.
High = 2	Buzzer high.

2.11.5 AsReaderP252BMemoryBank

Definitions	Descriptions
Reserved=0	Reserved memory Bank.
EPC = 1	EPC memory Bank.
TID = 2	TID memory Bank.
User = 3	User memory Bank.

2.11.6 AsReaderP252BQuerySession

Definitions	Descriptions
S0=0	Inventoried S0.
S1=1	Inventoried S1.
S2=2	Inventoried S2.
S3=3	Inventoried S3.

2.11.7 AsReaderP252BSessionFlag

Definitions	Descriptions
A=0	A only.
B=1	B only.
AB=2	A or B.

2.11.8 AsReaderP252BResultCode

Definitions	Descriptions
NoError = 0x0000	Succeed in result.
OtherError = 0x0001	An error has occurred due to unknown reason.
Undefined = 0x0002	An undefined error
MemoryOverrun = 0x0003	Accessing to memory out of range.
MemoryLocked = 0x0004	Memory is locked.
InsufficientPower = 0x000B	Battery power is low.
NonSpecificError = 0x000F	Not a specific error.
InOperation = 0xE000	In operation.
NotConnected = 0xE100	Not connected to device.
InvalidParameter = 0xE200	Invalid parameter transmitted.
NotSupportFirmware = 0xEE00	Unsupported firmware.
Timeout = 0xEFFF	Exceeded allowed accessing time.
HandleMismatch = 0xF001	Handle mismatch.
CRCError = 0xF002	CRC error on tag response.
CommandFormatError = 0xF007	Command format error.
OutOfRetries = 0xF009	Access password error, kill password error, and other errors.
OperationFailed = 0xFFFF	Operation failed.

2.11.9 AsReaderP252BScanMode

Definitions	Descriptions
RFIDScanMode = 0	RFID mode.
BarcodeScanMode = 1	Barcode mode.

2.11.10 AsReaderP252BTemperatureTagType

Definitions	Descriptions
Type_0= 0	The type of temperature tag is Magnus-S3.

2.11.11 AsReaderP252BBarcodeType

Definitions	Descriptions
BarcodeTypeNoRead	UNKNOWNTYPE
BarcodeTypeCode39	Code39
BarcodeTypeCode11	Code11
BarcodeTypeCodabar	Codabar
BarcodeTypeEAN13	EAN-13
BarcodeTypeCode128	Code128
BarcodeTypeEAN13With2Supps	EAN-13 with 2 Supps
BarcodeTypeIndustrial2Of5	Industrial 2 of 5
BarcodeTypeEAN13With5Supps	EAN-13 with 5 Supps
BarcodeTypeIATA2Of5	IATA 2 of 5
BarcodeTypeMSI	MSI
BarcodeTypeInterleaved2Of5	Interleaved 2 of 5
BarcodeTypeEAN128	EAN-128
BarcodeTypeCode93	Code93
BarcodeTypeUPCE1	UPC-E1
BarcodeTypeUPCA	UPC-A
BarcodeTypeUPCE1With2Supps	UPC-E1 with 2 Supps
BarcodeTypeUPCE1With5Supps	UPC-E1 with 5 Supps
BarcodeTypeUPCAWith2Supps	UPC-A with 2 Supps
BarcodeTypeTriopticCode39	Trioptic Code39
BarcodeTypeUPCE0	UPC-E
BarcodeTypeBooklandEAN	Bookland EAN
BarcodeTypeUPCE0With2Supps	UPC-E with 2 Supps
BarcodeTypeCouponCode	Coupon Code
BarcodeTypeUPCE0With5Supps	UPC-E with 5 Supps
BarcodeTypeGS1DataBarLimitedRSSLimited	GS1 DataBar Limited (RSS-Limited)
BarcodeTypeEAN8	EAN-8
BarcodeTypeGS1DataBarRSS14	GS1 DataBar (RSS-14)
BarcodeTypeEAN8With2Supps	EAN-8 with 2 Supps
BarcodeTypeGS1DataBarExpandedRSSExpanded	GS1 DataBar Expanded (RSS-Expanded)
BarcodeTypeEAN8With5Supps	EAN-8 with 5 Supps
BarcodeTypeMatrix2Of5	Matrix 2 of 5
BarcodeTypeChinaPostChinese2Of5	China Post (Chinese 2 of 5)
BarcodeTypeCode32	Code32
BarcodeTypeUKPlessey	UK Plessey
BarcodeTypeISBT128	ISBT128
ParameterFNC3	Parameter (FNC3)
BarcodeTypePDF417	PDF417

BarcodeTypeAztec	Aztec
BarcodeTypeMicroPDF417	MicroPDF417
BarcodeTypeQR	QR
BarcodeTypeDataMatrix	DataMatrix
BarcodeTypeMicroQR	Micro QR
BarcodeTypeHanXinCode	HanXin Code
BarcodeTypeMaxicode	Maxicode
BarcodeTypeITF14	ITF-14
BarcodeTypeITF6	ITF-6
BarcodeTypeAIM128	AIM 128
BarcodeTypeISSN	ISSN
BarcodeTypeISBN	ISBN
BarcodeTypeGS1Databar	GS1-Databar

AsReaderP252B SDK

Android SDK Reference Guide

Mar. 2024 1st Edition

AsReader Inc.

111 SW 5th Ave., Ste 3150

Portland, OR 97204-3656　U.S.A.

Tel.: (503) 770-2777 x102

Asterisk Inc.

AsTech Osaka Building 5F,

2-2-1, Kikawa-nishi,

Yodogawa-ku, Osaka, 532-0013, JAPAN