Android SDK

Developer's Guide

Connection switching management

Ver 1.00.00

COPYWRIGHT

Apulsetech is the manufacturer of Apulsetech RFID handheld computers.
This document and related software in this device are protected by international copyright laws.
Any part of this document may not be reproduced, removed or used in any form by any means, without permission in writing from Apulsetech.
The contents in this manual are subject to change without prior notice.

Apulsetech and α811 are registered trademarks of Apulsetech Co., Ltd., all other trademarks and copyrights are property of their respective owners.

The software is provided for user understanding of how to use the device and application development. All software, including firmware, is on a licensed basis.
No right to copy a licensed program in whole or in part is granted, to other devices except as permitted under copyright law.

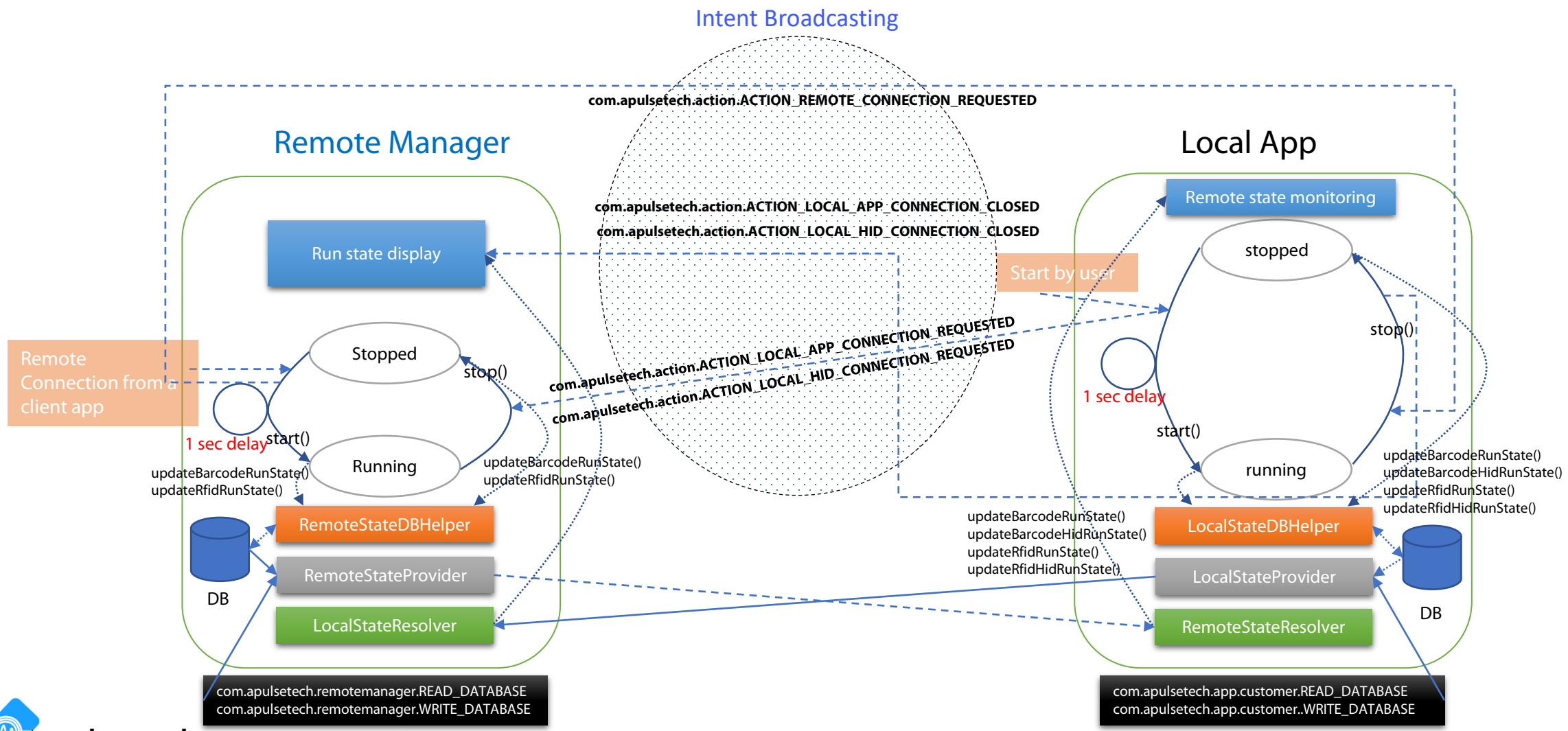
Apulsetech Co., Ltd.
C-1211, 60, Haan-ro,
Gwangmyeong-si,
Gyeonggi-do,
Republic of Korea

http://www.apulsetech.com

REVISION HISTORY

Change	Date	Description
Ver.1.00.00	2020/03/15	1st Release

CONNECTION SWITCHING MANAGEMENT



IMPLEMENATION ON THE CUSTOM DEMO APP

Mandatory

1. Add code lines to send a '*com.apulsetech.action.ACTION_LOCAL_APP_CONNECTION_REQUESTED*' or '*com.apulsetech.action.ACTION_LOCAL_HID_CONNECTION_REQUESTD*' intent to the remote manager app before the starting code of a barcode scanner or RFID reader according to the type of custom app.

```
Intent localAppIntent = new Intent(CustomContext.ACTION_LOCAL_APP_CONNECTION_REQUESTED);
localAppIntent.putExtra(CustomContext.EXTRA_MODULE, Module.BARCODE);
sendBroadcast(localAppIntent);
mHandler.sendEmptyMessageDelayed(MSG_CLIENT_DELAY_CONNECT, DemoConfig.CONFIG_INT_FORCE_CONNECT_DELAY);
```

2. Add code lines to send a '*com.apulsetech.action.ACTION_LOCAL_APP_CONNECTION_CLOSED*' or '*com.apulsetech.action.ACTION_LOCAL_HID_CONNECTION_CLOSED*' intent to the remote manager app after stopping code of a barcode scanner or RFID reader according to the type of custom app. For more detail, please refer to '*BarcodeMainActivity.java*' or '*RfidMainActivity.java*' in the sdk demo projects.

```
Intent localAppIntent = new Intent(CustomContext.ACTION_LOCAL_APP_CONNECTION_CLOSED);
localAppIntent.putExtra(CustomContext.EXTRA_MODULE, Module.BARCODE);
sendBroadcast(localAppIntent);
```

3. Create a broadcast receiver to handle a '*com.apulsetech.action.ACTION_REMOTE_CONNECTION_REQUESTED*' intent.
4. Add proper code lines to close a barcode scanner and RFID reader or app itself. For more detail, please refer to '*DemoApplication.java*'. In the sdk demo projects.

```
registerReceiver(mApplicationReceiver,
        new IntentFilter(CustomContext.ACTION_REMOTE_CONNECTION_REQUESTED));
```

IMPLEMENATION ON THE CUSTOM DEMO APP

```
private BroadcastReceiver mApplicationReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        LogUtil.log(LogUtil.LV_D, D, TAG, "onReceive()");

        int barcodeRunState = mStateDBHelper.getBarcodeRunState();
        int barcodeHidRunState = mStateDBHelper.getBarcodeHidRunState();
        int rfidRunState = mStateDBHelper.getRfidRunState();
        int rfidHidRunState = mStateDBHelper.getRfidHidRunState();
        boolean barcodeRunning = barcodeRunState != 0;
        boolean barcodeHidRunning = barcodeHidRunState != 0;
        boolean rfidRunning = rfidRunState != 0;
        boolean rfidHidRunning = rfidHidRunState != 0;
        boolean runState = barcodeRunning || rfidRunning;
        boolean hidRunState = barcodeHidRunning || rfidHidRunning;

        LogUtil.log(LogUtil.LV_D, D, TAG,
                "runState=" + runState
                        + " barcdoeRunState=" + barcodeRunState
                        + " rfidRunState=" + rfidRunState);

        if (runState) {
            LogUtil.log(LogUtil.LV_D, D, TAG, "Close the function module.");
            Intent baseActivityIntent = new Intent(DemoApplication.this, FunctionSelectorActivity.class);
            baseActivityIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(baseActivityIntent);
        }

        if (hidRunState) {
            if (barcodeHidRunning) {
                stopService(new Intent(DemoApplication.this, BarcodeHidService.class));
            }

            if (rfidHidRunning) {
                stopService(new Intent(DemoApplication.this, RfidHidService.class));
            }
        }
    }
};
```

IMPLEMENATION ON THE CUSTOM DEMO APP

Optional(for run state monitoring and fine control)

1. Add '*updateBarcodeRunState()*' and '*updateBarcodeHidRunState()*' methods for a barcode scanner, and '*updateRfidRunState()*' and '*updateRfidHidRunState()*' methods for a RFID reader to manage the run state of each function in DB. And to get current run state of each function, you can use '*getBarcodeRunState()*' and '*getBarcodeHidRunState()*' methods of the *LocalStateDBHelper* class for a barcode scanner, and '*getRfidRunState()*' and '*getRfidHidRunState()*' for a RFID reader.

```
mStateDBHelper = new LocalStateDBHelper(this);
```

```
mStateDBHelper.updateBarcodeRunState(LocalStateDBHelper.VALUE_RUNNING);
```

```
mStateDBHelper.updateBarcodeRunState(LocalStateDBHelper.VALUE_STOPPED);
```

2. Add required permissions('*com.apulsetech.app.customer.READ_DATABASE*' and '*com.apulsetech.app.customer.WRITE_DATABASE*') and declare a local content provider in '*AndroidManifest.xml*'. For more detail, please refer to the 'AndroidManifest.xml' file in sdk demo projects.

```
<uses-permission android:name="com.apulsetech.remotemanager.READ_DATABASE"/>
<uses-permission android:name="com.apulsetech.remotemanager.WRITE_DATABASE"/>

<permission android:name="com.apulsetech.app.customer.READ_DATABASE" android:protectionLevel="normal" />
<permission android:name="com.apulsetech.app.customer.WRITE_DATABASE" android:protectionLevel="normal" />
```

```
<provider android:name="com.apulsetech.lib.provider.LocalStateProvider"
    android:authorities="com.apulsetech.app.unified.LocalStateProvider"
    android:exported="true"
    android:readPermission="com.apulsetech.app.customer.READ_DATABASE"
    android:writePermission="com.apulsetech.app.customer.WRITE_DATABASE"/>
```

IMPLEMENATION ON THE CUSTOM DEMO APP

3. To get the run state of the remote connection from a remote client, you can use a '*RemoteStateResolver.isRemoteScannerRunning()*' method for a barcode scanner and '*RemoteStateResolver.isRemoteReaderRunning()*' for a RFID reader respectively. For more detail, please refer to demo app projects in the sdk package.

```
RemoteStateResolver.isRemoteScannerRunning(FunctionSelectorActivity.this)
```

```
RemoteStateResolver.isRemoteReaderRunning(FunctionSelectorActivity.this)
```