Accessory Connect Agent user manual

Version History

Version	Author	Date	Description
1.0.11	lihuihui	2023-03-23	This section describes how to access a service

1 Purpose of writing

The AccessoryConnectAgent service is mainly used for data interaction between D22 and Q3 by sending Intent

2 Usage scenario

In the subsequent use, the customer can exchange information through the AIDL interface provided by the Service

For example:

D22 sends Intent through AIDL to notify Q3 of the transaction amount and open the card swiping device;

After the transaction is completed, Q3 sends Intent through AIDL to notify D22 of the transaction result and prints receipts

3 Interface description

This application uses the system background service

When only one Q3 is connected to D22, it automatically establishes a link with the Q3 and runs without interface

When more than one Q3 device is connected, the serial number of Q3 device will be displayed in the pop-up window during startup or initial service installation. Customers can select one of the Q3 devices to connect. If you want to re-select the device, you need to restart D22

4 Service Access Guide

The service provides AIDL files (java) externally. Apps that need to use AccessoryConnectAgent for data interaction need to add AIDL files to the project

```
package com.smartpos.accessoryagent.aidl;

interface IRemoteAccessoryApi{

    String remoteIntent(String intentJsonString);

}
```

The mandatory type of the interface is String, and the parameter format is JSON. Customers interact with data through this interface

4.1 Service start

Service information:

package="com.smartpos.accessoryagent"

service="com.smartpos.accessoryagent.service.RemoteAccessoryApiService"

The app that needs to use AccessoryConnectAgent for data interaction needs to start the service through bind

service usage procedure:

- Start the Service using the bindService(Intent, serviceConnection, flags) method

Intent: Identifies the service to be connected. The Intent must specify an explicit component name

serviceConnection: Receives information when the service starts and stops. This must be a valid serviceConnection object. It cannot be empty

flags: Bind operation options

- If the service is not needed any more, call the unbindService(serviceConnection) method to stop the service

See service binding details:

https://developer.android.google.cn/reference/android/content/Context

example(kotlin):

```
val intent = Intent()
intent.component =
        ComponentName("com.smartpos.accessoryagent","com.smartpos.accessoryagent.service.RemoteAccessoryApiService")
bindService(intent, mServiceConnection, BIND_AUTO_CREATE)
```

4.2 Data sending (startup of target app)

The AIDL interface defines a String with a JSON parameter format and sends data with remoteIntent(json)

JSON contains the fields required by the Intent object to start the target

app. You can start the target app by setting the following fields, which cannot be empty

- "packageName": Represents the application package name
- "className": Represents the class name in the application package
- "action": Represents the action of the application
- "putExtra": Represents extended data, which can be set based on service requirements
- "flags": Set flags to control how to handle Intents, and flags to the startup mode

Intent flags:

- FLAG_ACTIVITY_SINGLE_TOP = 0x20000000: It starts the activity directly if it exists at the top of the stack, and recreates it if it does not exist
- FLAG_ACTIVITY_NEW_TASK = 0x10000000: Non-activity When you start an activity, you create a new stack to hold the activity
- FLAG_ACTIVITY_CLEAR_TOP = 0x04000000: The activity is not recreated if it exists on the stack
- FLAG_ACTIVITY_NO_HISTORY = 0x40000000: The activity is not pushed onto the stack

For other flags, see intent.setFlags()

https://developer.android.google.cn/reference/android/content/Intent#setFlags(int)

JSON example(kotlin):

```
var intent = "{
    "packageName":"com.smartpos.todemo",
    "className":"com.smartpos.todemo.MainActivity",
    "action":"android.intent.action.MAIN",
    "putExtra":{
    "salename":"sale",
    "payments":"card",
    "payamount":"1200"
},
    "flags":0x10000000
} "
iRemoteAccessoryApi?.remoteIntent(intent)
```

After receiving the data, the AccessoryConnectAgent service parses the JSON, creates the Intent object, starts the corresponding APP activity according to the JSON content, and transmits the data

4.3 Data Reception

The target app's activity gets the data it passes with the intent

example(kotlin):

```
// Get the package name:
packageName = intent?.component?.packageName
//Get the class name::
className = intent?.component?.className
//Get the action
action = intent?.action
//Get extended data:
putExtra = intent?.getStringExtra("salename")
```

5 Service installation

SetUsbAttrs.apk： The command is used to enable or disable the AccessoryConnection Agent service

persist.wp.usbchannel=1 The AccessoryConnectAgent service is enabled. Other values are disabled. The Settings take effect after the machine is successfully restarted

- Install SetUsb.apk on the D22 and Q3 devices, respectively
- OPEN USB Enables the AccessoryConnectAgent service

Restart the machine

AccessoryConnectionAgent.apk: The service developed this time

Install this service in D22 and Q3, turn on usbchannel, restart the device and start communication