CardSystem interface instructions for customized development

1. Customers who want to developing some functions or special requests by using xixun android system controller, please refer to this instructions; use AIDL provided by Android binding services to keep communication between processes.

2. Files and folders usage

2.1, folder “AidlDemo”is part of demo of how to use aidl interface.

2.2, file “CardService.aidl” is aidl configuration file, contains all method names for all of services that we provided.

2.3, file “**xixun_card_settings_1.2.1.jar**” is customized development service jar package, need to import it into Android project.

2.4,file “task_dto5.jar”is customized development scheduled task data type jar package, if need customized scheduled tasks, like brightness, volume, screen on/off in schedule, then need to introduce this package, and use it together with “**xixun_card_settings_1.2.1.jar**”

3. NOTES

3.1 Should communicate in sub thread when using aidl interface;

3.2 Aidl bind service adopt asynchronous operation, so need to judge if it is empty after getting the bing service. If not empty, then can use this interface; if empty, then need to acquire it again.

4. Modified logs

2018.05.18:

Added RealtimeServer sending broadcast :

Action： "xixun.intent.action.SET_REALTIME_SERVER"

intent.getStringExtra(“server”);

2018.04.08:

Added sending broadcast when system power off:

Action： "xixun.intent.action.POWER_OFF"

2018.06.13:

Added GPIO output interface, Need CardSystem version 4.9.9 or above, use AIDL interface executeJsonCommand to transfer data:

setup command data format：

```
{
    "id": "String",   // character stringID，can put random character string
    "_type": "OperationGPIO",
    “method”: 1,    // 0:config GPIO work mode；1：read GPIO；2：write GPIO
    "group": 'B',    // if no can leave empty
    "num": “10”,    // GPIO port number
    "value": "1",    // as write GPIO,1 is output high；
                     // as config work mode, 1 is config as output mode, 0 is config input mode
}
```

If error, return JSON character string{"_type":"Error","errorMessage": "xxxx"}，if success, return{"_type":"Success"}

2017.03.01

Temperature and humidity broadcast(send this broadcast when get sensor values)Action： “xixun.intent.action.TEMPERATURE_HUMIDITY”

```
intent.getFloatExtra("temperature", 0f);
intent.getFloatExtra("humidity", 0f);
intent.getFloatExtra("noise", 0f);
intent.getIntExtra("pm2.5",0);
intent.getIntExtra("pm10", 0);
intent.getFloatExtra("windSpeed", 0f);
intent.getIntExtra("windDirection",0);
```

2017.02.14

Get RF mode time（system will send broadcast at the same time when get RF mode time）

Action： "xixun.intent.action.RF_TIME”

Get method：long rfTime = intent.getLongExtra("time", 0);

2016.12.08

Send GPS information broadcast (for customized development to receive GPS information)

```
Intent intent = new Intent("com.xixun.joey.gpsinfo");
intent.putExtra("time", gps.getTime());          // time，data type：long
intent.putExtra("latitude", gps.getLatitude()); // latitude, data type：double
intent.putExtra("longitude", gps.getLongitude());// longitude, data type：double
intent.putExtra("satelliteNumber", gps.getSatelliteNumber());         // satellite numbers，data type：int
intent.putExtra("accuracy", gps.getAccuracy());       // accuracy, data type:float
intent.putExtra("altitude", gps.getAltitude());  // altitude, data type:float
intent.putExtra("speed", gps.getSpeed());     //speed, data type:float
intent.putExtra("azimuth", gps.getAzimuth());          // azimuth, data type:float
context.sendBroadcast(intent);
```

2016.09.21：

Added APN config interface, send data by broadcast, data as following：

```
Intent intent = new Intent("xixun.intent.action.XIXUN_SET_APN");
intent.putExtra("name", "China Unicom");
```

intent.putExtra("apn", "3gnet");
intent.putExtra("user", "");
intent.putExtra("password", "");
this.sendBroadcast(intent);

2016.1.08:

Add new important interfaces, detect network detect address, to sure smoothly detecting network IP address, using AIDL interface exectueJsonCommand to transferring data:

config command data type: {
"_type": "SetWatchingAddress"
"address": “IP or Domain" // IP address or domain name
}

If error, will return JSON character string{"_type":"Error","errorMessage": "xxxx"}, if no errorm will return {"_type":"Success"}

Get command data type:
{
"_type": "GetWatchingAddress",
"address": "IP or domain", //IPaddress or domain name

If error after receiving command, then will return JSON character string{"_type":"Error","errorMessage": "xxxx"}if no errror will return {"_type":"DataCallback","address": "IP or domain"}