

GpService.aidl

Package : com.gprinter.aidl

File Name: GpService.aidl

PrinterId: Range 0、1、2

Reference: Printer ID, plug-in Gplink simultaneously connect 3 printers available: Printer001, Printer002, Printer003, numbered 0 、 1 、 2

int openPort(**int** PrinterId, **int** PortType, **String** DeviceName, **int** PortNumber)

Function: open the port, connection process is unsynchronized, register "action.connect.status" to receive port connection status

GpDevice. STATE_NONE = 0; // disconnect

GpDevice. STATE_LISTEN = 1; // listening state

GpDevice. STATE_CONNECTING = 2; // connecting

GpDevice. STATE_CONNECTED = 3; // connected

GpDevice. STATE_INVALID_PRINTER = 4; // invalid printer

GpDevice. STATE_VALID_PRINTER = 5; // valid printer

PrinterId: Printer ID

PortType: **Port type**

PortParameters.USB = 2

PortParameters.ETHERNET = 3

PortParameters.BLUETOOTH = 4

DeviceName: **Device name**

If USB port, it would be USB Device Name

If Ethernet port, it would be IP address

If Bluetooth port, it would be Bluetooth Mac address

PortNumber: **Port number**

If USB port, it would be 0

If Ethernet port, it would be port number, 9100 as normal

If Bluetooth port, it would be 0

Return value: error value, if connected then return ERROR_CODE.[DEVICE_ALREADY_OPEN](#)

Error Value Reference:

enum ERROR_CODE {

[SUCCESS](#), // success

[FAILED](#), // failed

[TIMEOUT](#), // timeout

[INVALID_DEVICE_PARAMETERS](#), // invalid device parameter

[DEVICE_ALREADY_OPEN](#), // device already open

[INVALID_PORT_NUMBER](#), // invalid port number

[INVALID_IP_ADDRESS](#), // invalid IP address

[INVALID_CALLBACK_OBJECT](#), // Invalid callback object

[BLUETOOTH_IS_NOT_SUPPORT](#), // Bluetooth is not support

[OPEN_BLUETOOTH](#), // open Bluetooth

[PORT_IS_NOT_OPEN](#), // port is not open

[INVALID_BLUETOOTH_ADDRESS](#), // invalid Bluetooth address

[PORT_IS_DISCONNECT](#) // port is disconnect

}

int closePort(**int** PrinterId)

Function: close the port
Parameter: PrinterId: Printer ID
Return value: error value
Reference: Error Value Reference

int getPrinterConnectStatus(**int** PrinterId)

Function: get printer connection status
Parameter: PrinterId: Printer ID
Return value: status value

```
GpDevice. STATE_NONE = 0;    // disconnect  
GpDevice. STATE_LISTEN = 1;  // listening state  
GpDevice. STATE_CONNECTING = 2; // connecting  
GpDevice. STATE_CONNECTED = 3; // connected
```

int printTestPage(**int** PrinterId)

Function: print test page
Parameter: PrinterId: Printer ID
Return value: error value
Reference: Error Value Reference

void queryPrinterStatus(**int** PrinterId, **int** Timesout, **void** requestCode)

Function: query printer status
Parameter: PrinterId: Printer ID
Timesout: receiving timeout ms
Set 500-1000ms timeout for Bluetooth version because of time lag between transmitting command and returning the status data, it depends on device and OS environment.
Set 100-500ms timeout for USB or Wi-Fi version because of returning the data more quickly by USB or WIFI.
Return value: none
Receiving method: return the status via broadcast
action of broadcast-> GpCom.ACTION_DEVICE_REAL_STATUS
extra of broadcast:
int requestCode =
intent.getIntExtra(GpCom.EXTRA_PRINTER_REQUEST_CODE, -1);
int status =
intent.getIntExtra(GpCom.EXTRA_PRINTER_REAL_STATUS, 16);

Status value reference:

```
STATE_NO_ERR = 0; // in normal  
STATE_OFFLINE = 0x1; // offline  
STATE_PAPER_ERR = 0x2; // out of paper  
STATE_COVER_OPEN = 0x4; // cover open  
STATE_ERR_OCCURS = 0x8; // overheating, error occurs  
Refer to broadcast receiver of MainActivity for more details.
```

int getPrinterCommandType(**int** PrinterId)

Function: get printer command type
Parameter: PrinterId: Printer ID
Return value: printer command type

```
ESC_COMMAND = 0;  
TSC_COMMAND = 1;
```

int sendEscCommand(**int** PrinterId,String b64)

Function: send ESC command; make sure the printer is in receipt mode when sending this command, otherwise invalid.

Parameter: PrinterId: Printer ID

String b64: receipt content data

Return value: error value is same as ERROR_CODE

int sendTscCommand(**int** PrinterId,String b64)

Function: send TSC command; make sure the printer is in label mode when sending this command, otherwise invalid.

Parameter: PrinterId: Printer ID

String b64: label content data

Return value: error value is same as ERROR_CODE

void isUserExperience(**boolean** userExperience)

Function: Participate in user experience

Parameter: userExperience

Return value: void