MFPTP 传输协议接口说明

V1.0	Released on March 1th, 2015, by Wangshiyou

制订：

审核：

标准化：

批准：

初始接口

MFPTP(String mfptpName,boolean openLog)

实例化对象 MFPTP

参数：

mfptpName - the user define name

openLog - MFPTP log switch

返回值：空

基础接口

getMfptpVerNo()

获取当前 MFPTP 的协议版本信息

返回值：byte

网络接口

getMfptpSocketStatus()

获取当前 MFPTP 协议的网络状态

返回值：整形

0 : MFPTP_SOCKET_NOT_CONNECT

1 : MFPTP_SOCKET_DISCONNECT

2 : MFPTP_SOCKET_CONNECTING

3 : MFPTP_SOCKET_CONNECT_FAILED

4 : MFPTP_SOCKET_CLOSE

5 : MFPTP_SOCKET_WRITE_ERR

6 : MFPTP_SOCKET_READ_ERR

7 : MFPTP_SOCKET_CONNECTED

8 : MFPTP_SOCKET_WRITE

9 : MFPTP_SOCKET_READ

connect(String url, int port, boolean false)

网络连接接口

参数：

url - the url of server.

port - the port of server port.

true - need resume transfer

返回值：空

close()

关闭当前 MFPTP 的服务器连接

返回值：boolean

send()

发送当前数据包

默认无压缩、无加密、push 操作

返回值：boolean

send(byte compression, byte encryption, byte socketType)

发送当前数据包

参数:

compression - the compression of send data. 0 : no compression 1 : zip 2 : gzip 3 : lz4

encryption - the encryption of send data. 0 : no encryption 1 : rsa 2 : dsa 3 : aes

socketType - the socketType of send data. 0 : pair 1 : pub 2 : sub 3 : req 4 : rep 5 : dealer 6 : router 7 : pull 8 : push

返回值：boolean

数据接口

setSendFrame(boolean newDataBody, byte[] sendFrame)

设置发送帧数据

参数：

newDataBody - is current dataBody over.

sendFrame - current frame data.

返回值：boolean

cleanReceiveData()

清除所有接收数据

返回值：空

回调接口

setProtocolEventHandler(MFPTPInterface callBack, int eventType)

设置协议事件回调函数

参数：

callBack - protocol user call back function.

eventType - protocol type. MFPTP_SOCKET_EVENT_NOTIFY_IND : 1

权限设置

```
<uses-permission android:name="android.permission.INTERNET" />
```

例子

```
private MFPTP mfptpSend = new MFPTP("send",true);
private MFPTP mfptpRecieve = new MFPTP("receive",true);

public class MfptpCallback1 implements MFPTPInterface {
        @Override
        public void socketNotify(final String mfptpName, final int socketStatus,
                        final MFPTPData mfptpData) {
                int i;
                String test = "wsy test ";
                byte[] sendBuffer = null;
                switch (socketStatus) {

                case MFPTP.MFPTP_SOCKET_NOT_CONNECT:
                        break;
                case MFPTP.MFPTP_SOCKET_DISCONNECT:
                        break;
                case MFPTP.MFPTP_SOCKET_CONNECTING:
                        break;
                case MFPTP.MFPTP_SOCKET_CONNECT_FAILED:
                        break;
                case MFPTP.MFPTP_SOCKET_CLOSE:
                        break;
                case MFPTP.MFPTP_SOCKET_WRITE_ERR:
                        break;
                case MFPTP.MFPTP_SOCKET_READ_ERR:
                        break;
                case MFPTP.MFPTP_SOCKET_CONNECTED:
                        for (i = 0; i < 10; i++) {
                                test = "wsy send txt and picture : " + i + "\r\n";
                                sendBuffer = test.getBytes();
                                mfptpSend.setSendFrame(true, sendBuffer);
                                InputStream inputStream1 = getResources()
                                                .openRawResource(R.raw.image_longlong);
                                InputStream inputStream2 = getResources()
                                                .openRawResource(R.raw.poweroff_confirm);
                                byte[] bbyte1 = null;
```

```
                byte[] bbyte2 = null;
                try {
                    bbyte1 = input2byte(inputStream1);
                    bbyte2 = input2byte(inputStream2);
                } catch (IOException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                }

                mfptpSend.setSendFrame(true, bbyte1);
                mfptpSend.setSendFrame(false, bbyte2);
                test = "wsy receive txt and picture : " + i + "\r\n";
                sendBuffer = test.getBytes();
                mfptpRecieve.setSendFrame(false, sendBuffer);

                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                if (mfptpName.equals("send")) {
                    mfptpSend.send();
                }
                if (mfptpName.equals("receive")) {
                    mfptpRecieve.send((byte) 0, (byte) 0, (byte) 7);
                }
                MFPTPLogUtil.i("2015.01.27 wsy createMFPTPSend test =    "
                        + test);
            }
            break;
        case MFPTP.MFPTP_SOCKET_WRITE:
            break;
        case MFPTP.MFPTP_SOCKET_READ:
            MFPTPLogUtil.i("2015.01.27 wsy socketNotify MFPTP_SOCKET_READ");
            break;
        default:
            break;
        }
    }
}

main()
{
```

```
        mfptpSend.connect("192.168.11.249", 7777, true);
        mfptpRecieve.connect("192.168.11.249", 8888, false);
        MfptpCallback1 callBack = new MfptpCallback1();
        mfptpSend.setProtocolEventHandler(callBack, 1);
    }
```