Android SDK

Specification Version 2.1.1

NeoLAB Convergence Inc.

Revision History

Ver	Date	Contents	
1.40	04-May, 15	Added Offline data	ARM
1.50	30-Nov, 15	Connection procedure for Multi app	LMS
2.00	13-Jun, 16	Revised to SDK 2.0	LMS
2.10	07-Jul, 16	Separate IPenDotListerner from onReceiveDot() of IPenMsgListener	LMS
2.11	22-Feb, 17	Bluetooth LE	CJY

Neo Smartpen Android SDK

Contents

Neo Smartpen Android SDK

1. About this document

This document defines the communication protocol between neo smartpen and apps. This protocol is not compatible with version 1.0, if the app needs to support NWP-F110 and NWP-F120, Protocol Ver 1.0 and Ver 2.0 have to be implemented.

Model Name	Pen Name	Protocol Version
NWP-F110	N2	1.0
NWP-F120	N/A	2.x
NWP-F50	N/A	2.x

2. Preface

Please prepare Android development environment(http://developer.android.com/sdk/index.html)

- Eclipse IDE (ADT) or android studio
- android SDK(min SDK 4.4-KitKat)

3. Add Library

nasdk_v[x.x].jar has to be copied into user's android project library

4. AndroidManifest.xml setting

Add manifest permission code to your android project.

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

5. Class summary

IPenCtrl	Connect Pen via Bluetooth
IPenMsgListener	Receive PenMsg Data from the pen

IPenDotListener	Receive Ncode data from the pen
PenMsg	Data structure
PenMsgType	A type of incoming data from the pen
Const/JsonTag	Definition of data information from the pen

6. register/unregister Broadcastreceiver

When pen controller instance initially created, context is registered by
setContext(), in order to register Broadcastreceiver, execute the below method
If Penctrl.registerBroadcastBTDuplicate () executes, Broadcastreceiver is
registered inside SDK.
If Penctrl.unregisterBroadcastBTDuplicate () executes, Broadcastreceiver is
unregistered inside SDK.

```java
public class NeoNoteApplication extends Application {
…
@Override
public void onCreate() {
   iPenCtrl = PenCtrl.getInstance();
   iPenCtrl.setContext(getApplicationContext())
   iPenCtrl.registerBroadcastBTDuplicate();
…
}

@Override
public void onTerminate ()
{
    iPenCtrl.unregisterBroadcastBTDuplicate();
   super.onTerminate();
}
```

7. Get started Pen controller

All the function(such as connection, disconnection, stroke transmission) are
implemented in pen controller class.

```java
public class MainActivity extends Activity implements IPenMsgListener
{
        private IPenCtrl iPenCtrl;

        @Override
        protected void onCreate(Bundle savedInstanceState)
        {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);
                …
        // create pen controller instance
                iPenCtrl = PenCtrl.getInstance();

                // Start pen controller
```

Neo Smartpen Android SDK

```
            iPenCtrl.startup();

            // register pen event callback listener
            iPenCtrl.setListener(this);
              // register pen ncode callback listener
              iPenCtrl.setDotListener(this);
    }
```

8. Pen connection

Pairing process is required before connecting pen. please refer
developer.android.com for pairing process and sample codes.

Connection function is called with parameters of pen mac_address and context.

```
String mac_address = "9c7bd20001d4";
iPenCtrl.connect(mac_address);
```

9. Pen Acknowledge process

Pen controller must register callback(IPenMsgListener) for receive response.
And if pen controller method is executed, SDK transfer the PenMsg and Dot data
to callback as an asynchronous response.

In this document, it depicts the realtime data transfer and connection. Please
refer sample source code for more details.

step 1. Pen connection completion
step 2. Transmits password
step 3. Transmit use notes to pen by pen controller(IPenCtrl.reqAddUsingNote).
step 4. Exception process due to Bluetooth connections to other apps.


```
@Override
    public void onReceiveDot(Dot  dot) {
            // when connected, Ncode information is transfered as a Primitive type.
    }
    @Override
    public void onReceiveMessage(final PenMsg penMsg) {
            // data other than Ncode will be contained to a message object named PenMsg
            switch ( penMsg.penMsgType )
            {
                    // step 1. Upon Connection competion (authentication is not proceeded)
                    case PenMsgType.PEN_CONNECTION_SUCCESS:
                            break;
                    // step 2. Request pen's password (request right after
```

Neo Smartpen Android SDK

```
PEN_CONNECTION_SUCCESS)
                    // default:  "0000"
                    case PenMsgType.PASSWORD_REQUEST:
                        String password = "0000";
                        iPenCtrl.inputPassword( password );
                            break;
                    // step 3. Transmit use notes to pen by pen controller
                    // the setting for the page in use information is needed
                    // as well as section and owner in use
                    // care must be taken on these information,
case PenMsgType.PEN_AUTHORIZED:
                            iPenCtrl.reqAddUsingNote( USING_SECTION_ID,
USING_OWNER_ID);
                            break;

                    // step 4.  Connection failure, pen is connected to another app
                    //acquire the packageName of the  app in connection
                    //disconnect Bluetooth by acquired info., retry alert must be given to UI
flow
                       // in this Sample, this procedure is implemented by Toast
                    case PenMsgType.PEN_CONNECTION_FAILURE_BTDUPLICATE:
                        String connected_Appname = "";
                        try
                        {
                            JSONObject job = new JSONObject( content );
                            connected_Appname = job.getString("packageName");
                        }
                        catch ( JSONException e )
                        {
                            e.printStackTrace();
                        }
                            Util.showToast( this, String.format("The pen is currently
connected to %s app. If you want to proceed, please disconnect the pen from %s
app.",connected_Appname,connected_Appname));
                            break;
                                               .
                                               .
                                               .
                    }
            }
```
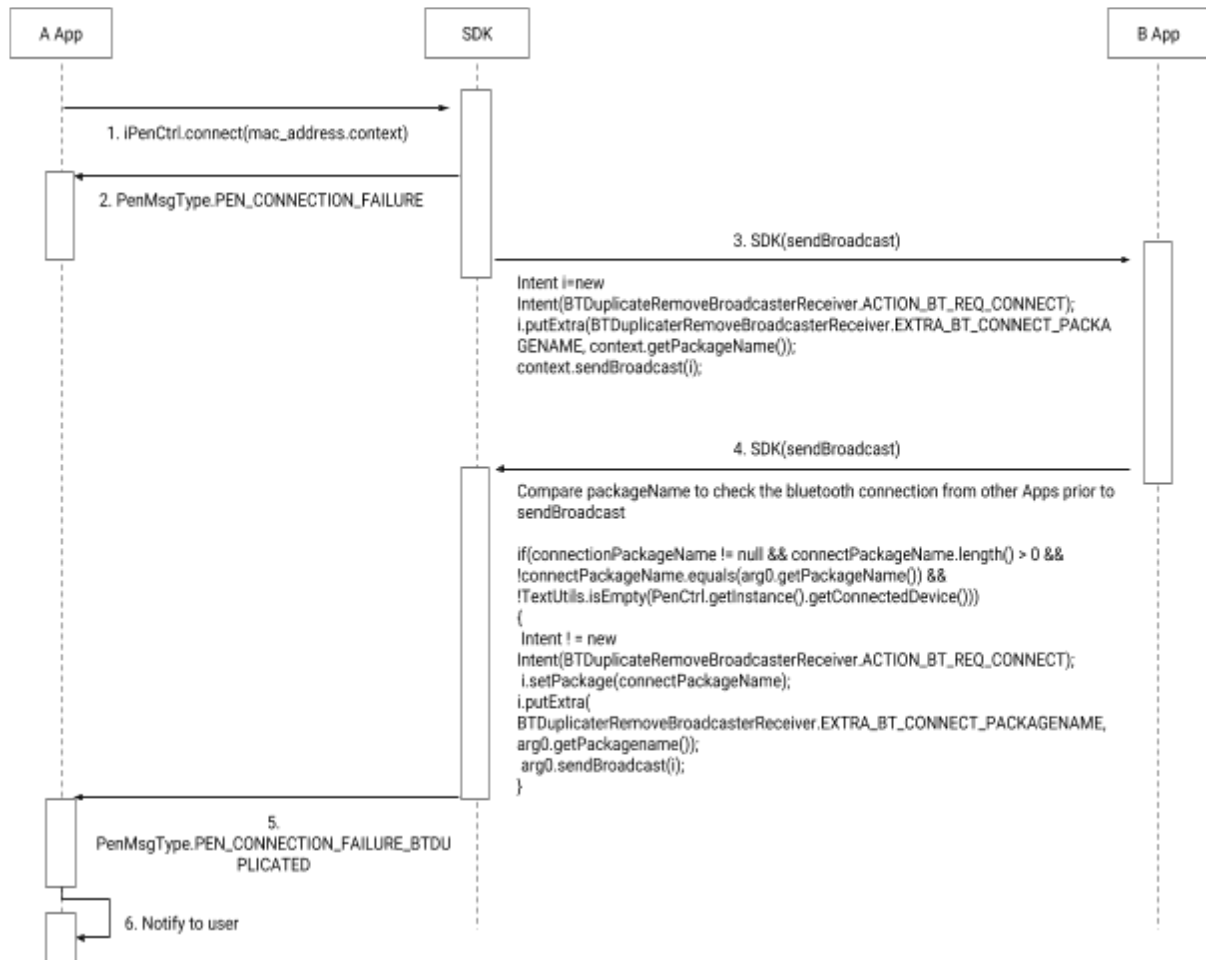
A sequential diagram is depicted as below.

Neo Smartpen Android SDK

10. onReceiveDot x, y, fx, fy

x, y : It coordinates of our dot code cell.(cell size is 2.371mm)
fx, fy : It is fractional part of our code.

Ex) (x + fx x 0.01) x 2.371 = millimeter unit

11. Notebook information

Note Type	NoteTitle	DX (ncode unit)	DY (ncode unit)	Width (ncode unit)	Height (ncode unit)
601	Pocket Note	5.40	5.45	35.26	62.59
602	Memo Note	5.40	5.45	35.26	62.59
603	Ring Note	5.52	5.41	63.46	88.88
605	FP Memo Pad	5.21	5.21	62.57	88.68
606	FP Original CEO	4.44	4.47	36.34	75.22
607	FP Original CL	4.33	4.46	62.81	91.09
608	Casual Planner	4.18	4.37	45.61	72.83

Neo Smartpen Android SDK

12 .nproj to ncode unit

Scale = 72 / 600 * 56 = 6.72

nproj size /scale = ncode size

ex) 500 height in nproj file, 500/Scale = 500/6.72 = 74.4 (ncode unit)

13. Offline Data Process

1. Request Offline data list : PenCtrl.reqOfflineDataList();

2. Receive Offline data list by callback

```java
@Override
public void onReceiveMessage( PenMsg penMsg )
{
  switch ( penMsg.penMsgType )
  {
    // Offline Data List response of the pen
  case PenMsgType.OFFLINE_DATA_NOTE_LIST:
    try
    {
      JSONArray list = new JSONArray( content );

      for ( int i = 0; i < list.length(); i++ )
      {
        JSONObject jobj = list.getJSONObject( i );

        int sectionId = jobj.getInt( JsonTag.INT_SECTION_ID );
        int ownerId = jobj.getInt( JsonTag.INT_OWNER_ID );
        int noteId = jobj.getInt( JsonTag.INT_NOTE_ID );
// request offline data
        Log.d( TAG, "offline(" + (i + 1) + ") note => sectionId : " + sectionId + ", ownerId
: " + ownerId + ", noteId : " + noteId );
      }
    }
    catch ( JSONException e )
    {
      e.printStackTrace();
    }
    Util.showToast( this, "offline data list is received." );

    break;
```

3. Reqeust ReqOfflineData
Request unit is a NoteBook unit,please make sure that OfflineData works asych
communication

PenCtrl. reqOfflineData(int sectionId, int ownerId, int noteId)

  4. Download information

Download start, finish, status, success/failure can be checked by a message of callback function.
If the message type is OFFLINE_DATA_FILE_CREATED, a download is completed.

Find offLineFile file path by JSON data(JsonTag.STRING_FILE_PATH).
Then if the offLineFile parsed, OffLineData work is completed.

```java
@Override
public void onReceiveMessage( PenMsg penMsg )
{
  switch ( penMsg.penMsgType )
  {
    // Messages for offline data transfer begins
  case PenMsgType.OFFLINE_DATA_SEND_START:
    break;

  // Offline data transfer completion
  case PenMsgType.OFFLINE_DATA_SEND_SUCCESS:
    break;

  // Offline data transfer failure
  case PenMsgType.OFFLINE_DATA_SEND_FAILURE:
    break;

  // Progress of the data transfer process offline
  case PenMsgType.OFFLINE_DATA_SEND_STATUS:
  {
    try
    {
      JSONObject job = new JSONObject( content );
      int total = job.getInt( JsonTag.INT_TOTAL_SIZE );
      int received = job.getInt( JsonTag.INT_RECEIVED_SIZE );
      Log.d( TAG, "offline data send status => total : " + total + ", progress : " + received
);
    }
    catch ( JSONException e )
    {
      e.printStackTrace();
    }
  }
    break;

  // When the file transfer process of the download offline
  case PenMsgType.OFFLINE_DATA_FILE_CREATED:
  {
    try
    {
      JSONObject job = new JSONObject( content );
      int sectionId = job.getInt( JsonTag.INT_SECTION_ID );
      int ownerId = job.getInt( JsonTag.INT_OWNER_ID );
      int noteId = job.getInt( JsonTag.INT_NOTE_ID );
      int pageId = job.getInt( JsonTag.INT_PAGE_ID );
      String filePath = job.getString( JsonTag.STRING_FILE_PATH );
      Log.d( TAG, "offline data file created => sectionId : " + sectionId + ", ownerId : " +
ownerId + ", noteId : " + noteId + ", pageId : " + pageId + " filePath : " + filePath );
    }
```

```
    catch ( JSONException e )
    {
      e.printStackTrace();
    }
  }
    break;

```

5. OffLineData Deletion

Received upon OffLineData를, the request of the file to pen can be made based
on OffLineData policy(if it is not deleted, the file information can be duplicated
when the next OffLineData request is made)

PenCtrl. removeOfflineData(int sectionId, int ownerId);

14. Protocol Ver 2.0 major changes in function

1. PenCap Off

A Method to on/off the power via pen cap is added

2. Stroke(Dot) Data added
More info added penTipType(pen/eraser) , tiltX/ tiltY(pen tilt), twist (pen skew)

3. Renderer class deletion
Stroke render is no longer supported in SDK
Rendering is provided as a sample code

4. PenHover mode
Hover mode can be turned on/off

5. Pen CALIBRATION
Pen calibration function is deleted from SDK

6. Changes in Pen Time value
Pen time value is referred by UTC

7. OffLineData

Offline date can be requested per page.
Unlike Protocol ver 1.0, Protocol ver 2.0 has been removed step
OFFLINE_DATA_FILE_CREATED.
Thus the stroke data can be received by callback directly without file work.

8. FW Upgrade
The fw file you transfer to the SDK should not be compressed.

Neo Smartpen Android SDK

9. reqAddUsingNote

Every Method related reqAddUsingNote overrides data from Protocol specification 2.0. If the multiple type of notebooks are requested, it has to be requested at a time via reqAddUsingNote(ArrayList<UseNoteData> noteList) Method

If your app support the protocol 1.0 and 2.0 regarding firmware, please check the comparison table below.

Method deleted	Method added
public void upgradePen(File fwFile, String targetPath) public void upgradePen(File fwFile)	public void upgradePen2(File fwFile, String fwVersion);
public void removeOfflineData(int sectionId, int ownerId)	public void reqOfflineDataRemove(int sectionId, int ownerId, int[] noteIds)
	public void reqOfflineDataList(int sectionId, int ownerId);
	public void reqOfflineDataPageList(int sectionId, int ownerId, int noteId);
	public void setOffLineDataListener(IOfflineDataListener listener);
	public IOfflineDataListener getOffLineDataListener();
	public void reqOfflineData(int sectionId, int ownerId, int noteId, int[] pageIds)
	public void reqAddUsingNote(ArrayList<UseNoteData> noteList)
	public void reqSetupPenCapOnOff(boolean on)
	public void reqSetupPenHover (boolean on)
	public int getProtocolVersion()
public void onReceiveDot(int sectionId, int ownerId, int noteId, int pageId, int x, int y, int fx, int fy, int pressure, long timestamp, int type, int color);	public void onReceiveDot(Dot dot)

PenMsgType deleted	PenMsgType Changed	PenMsgType Modified
PEN_CALIBRATION_START		
PEN_CALIBRATION_FINISH		
	PEN_SETUP_PEN_CAP_ONOFF	
	PEN_SETUP_HOVER_ONOFF	
	PEN_SETUP_OFFLINEDATA_SAVE_ONOFF	
	EVENT_LOW_BATTERY JSON Data "battery"	
	EVENT_POWER_OFF JSON Data "poweroff_reason"	

Neo Smartpen Android SDK

OFFLINE_DATA_PAGE_LIST	
	PEN_STATUS JSON Data deleted "status" "timezone" "pen_tip_color" "acceleration_sensor_onoff" JSON Data added "pencap_onoff" "hover_mode" "offlinedata_save" "battery" the value 128 indicates charging\
	PASSWORD_SETUP_FAILURE JSON Data Added "retry_count" "reset_count"

15. Realtime stroke date in Protocol Ver 2.0

Due to the changes in protocol 2.0, IPenMsgListener is changed accordingly.

In Protocol 1.0
public void onReceiveDot(int sectionId, int ownerId, int noteId, int pageId, int x, int y, int fx, int fy, int pressure, long timestamp, int type, int color)

In Protocol 2.0
public void onReceiveDot(Dot dot)
Dot object includes the old data and new added data.
In protocol 1.0, the value of newly added data will be filled with initial value.

```
@Override
public void onReceiveDot( Dot dot )
{
  NLog.d( "onReceiveDot sectionId=" + dot.sectionId + ",ownerId=" + dot.ownerId +
";noteId=" + dot.noteId + ";pageId=" + dot.pageId + ";x=" + dot.x + ";x=" + dot.y +
";y=" + dot.fx + ";fy=" + dot.fy + ";pressure=" + dot.force + ";timestamp=" +
dot.timestamp + ";type=" + dot.dotType + ";color=" + dot.color + ";tiltX=" + dot.tiltX +
";tiltY=" + dot.tiltY + ";twist=" + dot.twist + ";penTipType=" + dot.penTipType );
  sendPenDotByBroadcast( dot );
}
```

16. Offline processing in Protocol Ver 2.0

1. Request the stored Offline data list from the pen
PenCtrl.reqOfflineDataList();

Neo Smartpen Android SDK

2. In case if you are requesting per page, request Offline data via PenCtrl.reqOfflineDataPageList(int sectionId, int ownerId, int noteId) based on the acquired sectioned, ownerId, noteId
3. Offline data is requested via PenCtrl.reqOfflineData()
4. Receive Offlinedata by a callback function

```
@Override
public void onReceiveOfflineStrokes ( Stroke[] strokes, int sectionId, int ownerId, int noteId )
{
    NLog.d( "onReceiveOfflineStrokes strokes="+strokes.length );
    Intent i = new Intent( Const.Broadcast.ACTION_OFFLINE_STROKES );
    i.putExtra( Const.Broadcast.OFFLINE_STROKES, strokes );
    context.sendBroadcast( i );
}
```

17. Bluetooth LE

Requirement

Andorid 5.0, API Level 21(Lollipop)

Supported Neo Smartpen

Pen that use protocol ver2 (ex. F-50, F-120, C200, D100 etc.)

How To Use

SPP and LE can be selected using setLeMode method added to iPenCtrl class.

If parameter is true, mode is LE. Otherwise is SPP.

```
iPenCtrl.setLeMode(true);
```

Other API of SPP and LE are all the same.

/End of Document/

Neo Smartpen Android SDK