iOS SDK

Specification Version 2.3.5

NeoLAB Convergence Inc.

Table of Contents

Revision history

17.DEC.2014	NISDK v1.0 and document release	L. Park/K.You
3.AUG.2015	NISDK v2.0.1 and document v1.1.2 release	L.Park
12.NOV.2015	NISDK v2.1 and document v1.2 release Add PUI (paper ui) for email Add delegate method for setting transferable note IDs Add delegate method for offline data note list count	L.Park
26.NOV.2015	NISDK v2.2 and document v1.3 release Add NJPenCommManagerNewPeripheral protocol and relative delegate method for discovered peripherals list during BT scan Add delegate method for notifying if a pen has been connected by other apps	L.Park
8.DEC.2015	NISDK v2.3 and document v1.4 release Add delegate method for getting a path to offline raw files before parsed Add command for setting transferable note IDs according to section and owner IDs from property list Add pen status none to BT scan Add “Change canvas color” to n2sample menu Add dot checker for offline data to “parseOfflineDots” delegate method in n2sample	L.Park
21.DEC.2015	NISDK v2.4 and document v1.5 release Replace delegate method for offline data note list count Add method for reading battery level and memory space of a pen Add index and additional description for header files	L.Park
12.JUL.2016	NISDK v2.5 and document v1.6 release Add drawing options for notebook background Add SDK2.0	L.Park
12.AUG.2016	NISDK v2.6 and document v1.7 release Add nproj file parsing for notebook and paper information Add zoom to page canvas view in Sample app Add 601, 602, 603 notebook nproj files Remove note_paper_info.plist and note_pui_info.plist Remove SDK2.0	L.Park
17.AUG.2016	NISDK v2.7 and document v1.8 release Add SDK2.0	L.Park
23.AUG.2016	NISDK v2.7.1 and document v1.9 release Add findApplicableSymbols delegate method Add reqAddUsingNote to NJPenCommManager replace activeNoteId:pageNum: delegate method with activeNoteId:pageNum: sectionId:ownerId:	L.Park
29.AUG.2016	NISDK v2.8 and document v2.0 release Add NJPage and NJStroke classes to n2sample and remove NJPage and NJStroke classes Add renderWithScale to NJStroke Change activeNoteId:pageNum: sectionId:ownerId:delegate method name into activeNoteIdForFirstStroke:pageNum: sectionId:ownerId: Change activeNoteId:pageNum: sectionId:ownerId:ForCreatedPage: delegate method name into activeNoteIdForFirstStroke:pageNum: sectionId:ownerId:	L.Park

5.SEP.2016	NISDK v2.8.1 and document v2.1 release Add findApplicableSymbolX:Y: to n2sample Add requestNewPageNotification API to NJPenCommManager	L.Park
30.SEP.2016	Document v2.2 release Add NPPaperManager description	L.Park
27.OCT.2016	NISDK v2.9 and Document v2.3 release Read paper information from either nproj file parsing or note_paper_info.plist file Add getNotePaperInfoForNotebook:pageNum:section:owner Add class method '(BOOL)section:owner:fromNotebookId:' to n2sample for customization and remove it from SDK	L.Park
4.NOV.2016	NISDK v2.9.1 and Document v2.3.1 release Add setBTIDForPenConnection API	L.Park
2.DEC.2016	NISDK v2.9.2 and Document v2.3.2 release Add reading BTLE MTU size from Core Bluetooth framework Add turning offline data saving mode on from mode off for F120/F50 Add FW download from server and FW update to n2sample	L.Park
16.FEB.2017	NISDK v2.9.6 release Add SUPPORT_PEN_LOCALSUBNAME feature to disable pen registration with BT ID functionality and BT local subname from Pen advertisement	L.Park
7.MAR.2017	NISDK v2.9.7 and Document v2.3.3 release Support D100	L.Park
22.NOV.2017	NISDK v2.9.9 and Document v2.3.4 release Add the explanation of [NJPenCommManager sharedInstance].isPenSDK2 in the document Add the example of [NJPenCommManager sharedInstance].isPenSDK2 in the sample app Add handling an exception case for SDK2.0 protocol in the framework	L.Park
21.SEP.2018	NISDK v2.9.15 and Document v2.3.5 release Add handling 0x6D pen NDAC error Add the delegate method, penCommMsg, for NDAC error message	L.Park

1. Overview

This application note is for Neo smartpen iOS SDK1.0 and SDK2.0 (NISDK framework ver2.9.7). Developing Neo smartpen application requires three entities. They are a Neo smartpen, a notebook with Ncode™ and iPhone.

1.1 Neo smartpen

While users write on notebooks with Neo smartpen as they do with normal pens, Neo smartpen reads coordinates on notebook and send them to iPhone. To send coordinates, Neo smartpen and iPhone are connected to each other via Bluetooth. The firmware version of Neo smartpen N2 (F110) should be above v1.05 or above and that of Neo smartpen N2 (F120), Neo smartpen M1 (F50) should be v1.01 or above for the NISDK framework ver2.9.7. Also, NISDK framework version should be above 2.9.7 for D100.

1.2 Notebook with Ncode™

Neo smartpen reads coordinates of strokes which a user is writing on a notebook. For Neo smartpen to read coordinates the notebook has Ncode™ printed on each pages. By reading the codes Neo smartpen retrieves some information. It involves,

- Coordinates of each stroke
- Note Type : Unique ID given to a specific notebook type.
- Page Number : Page number which the stroke is written on

1.3 iPhone

Neo smartpen can communicate with any devices equipped Bluetooth. This application note uses iPhone for its counterpart. More specifically they are connected via Bluetooth Low Energy protocol(BTLE). The iPhone in BTLE communication is "central" device and the Neo smartpen is "peripheral". Usual application may requires iPhone to be responsible for,

- Connecting/Disconnecting to Neo smartpen via BTLE
- Saving strokes in its storage
- Rendering the stroke

1.4 URL scheme

The URL of the app is “Neo Notes” for URL scheme. You can call the app from the app version 1.6.

2. SDK structure

The iOS SDK sources are grouped under n2base folder. There are 4 sub-folders under the folder.

Name	Description
penComm	Bluetooth communication and protocol parser
common	Strings, definitions
utils	Utility functions
background	pdf files to be used as notebook background images

2.1 penComm

Applications use NJPenCommManager to send data to Neo smartpen and control BTLE connection. It is designed to be used as singleton across the application. The best practice to get instance of the Class looks like below.

```
pencommManager = [NJPenCommManager sharedInstance];
```

NJPenCommManger handles BTLE protocol related communications such as scan, connecting, disconnecting, services, characteristics, etc.
NJPenCommParser handles protocol defined for communication between Neo smartpen and iPhone.

When you would like to know whether SDK2.0 is working or SDK1.0 is working, you should read the following flag. If it is YES, SDK2.0 is operating. This flag is decided according to Service UUID for SDK2.0 inside the framework.

[[NJPenCommManager sharedInstance] isPenSDK2]

However, if you would like to make your app connected with the Neo smartpen (peripheral object) collected during the scanning period on your app side, [NJPenCommManager sharedInstance].isPenSDK2 should be set into YES (SDK2.0) or NO (SDK1.0) on it, default NO(SDK1.0).

Please refer to NJViewController class, sample app for the example.

if you choose 'BT List' of the menu in the sample app, the app will commence scanning smartpens for 3 seconds and the method '(void)discoveredPeripheralsAndConnect' will be performed after 3 seconds. In this method, [NJPenCommManager sharedInstance].isPenSDK2 should be set according to the service uuid.

2.2 pageData

The activity writing on a notebook is delivered to an iOS device as a group of coordinate representing a dot. Each dot has x, y coordinate on the page of the notebook. Each group may have one or more dots. And we call the group of dots as stroke.
A stroke is defined as a set of dots sampled by Neo smartpen from the moment the smartpen touches the notebook page(Pen Down) to the moment the smartpen is pulled up to be separated from the paper(Pen Up). NJStroke represents a stroke.
While a user keeps writing on a page, there should be a lot of strokes delivered to an iOS device.
NJPage is container of strokes written on the same page.
NJPage and NJStroke should be implemented on application side for NISDK version 2.8 or above.
Please refer to n2sample app for more details.

3. Getting Started

This chapter covers what you need to go through in order to be able to start using the Neo smartpen N2 (F110, F120), Neo smartpen M1 (F50) SDK for iOS. The 4 easy steps are as follows:

- Step 1: Prerequisites
- Step 2: Prepare the N2 SDK for iOS
- Step 3: Add Neo smartpen N2 (F110, F120), Neo smartpen M1 (F50) SDK to your Xcode project
- Step 4: Start coding

3.1 Step 1 : Prerequisites

- OS X is required for all iOS development
- You need Xcode. If you don't have it, you can get it from the App Store.

Note: The N2 SDK v2.9.7 for iOS supports iOS 8.x and higher.

3.2 Step 2: Prepare Neo smartpen SDK for iOS

The Neo smartpen N2 (F110, F120), Neo smartpen M1 (F50) SDK for iOS is a framework which is called as "NISDK.framework".
You will be provided with NISDK.framework separately or you can get it from the path of the sample code, "n2sample/NISDK.framework"

3.3 Step 3: Add Neo smartpen SDK to your Xcode Project

- Add Neo smartpen N2 (F110, F120), Neo smartpen M1 (F50) SDK for iOS to your Xcode project
    - Create a new project in Xcode
    - Select the NISDK.framework and drag it into the "Frameworks" group in Xcode
- Add a framework (libz.dylib)

3.4 Step 4: Start coding

- Add Neo smartpen N2 (F110, F120), Neo smartpen M1 (F50) SDK for iOS header file
    - #import <NISDK/NISDK.h>

4. Using iOS SDK1.0 or SDK2.0

4.1 Controlling BTLE connection(connect/disconnect)

4.1.1 Connect/Disconnect

With an instance of NJPenCommManager,

- [pencommManager btStart]
  Scan Neo smartpen and connect to the smartpen found.
- [pencommManager btStartForPeripheralsList]
  Scan Neo smartpen to obtain peripheral list around.
- [pencommManager disConnect]
  Disconnect from Neo smartpen
- [pencommManager btStop]
  Stop scanning Neo smartpen.
- [pencommManager hasPenRegistered]
  a flag for whether or not smartpen was registered. If a smartpen is registered, it will be YES. If not, it will be NO.
- [pencommManager isPenConnected]
  a flag for whether or not smartpen was connected. If a smartpen is connected, it will be YES. If not, it will be NO.
- [pencommManager resetPenRegistration]
  Reset smartpen registration. It will disconnect with Neo smartpen and delete a uuid registered.
  Also, set [pencommManager hasPenRegistered] to NO.
- [pencommManager regUuid]
  Return a uuid string for a smartpen registered and connected
- NJPenCommManPenConnectionStatus penConnectStatus
  -. Show a smartpen connection status.
  -. typedef NS_ENUM (NSInteger, NJPenCommManPenConnectionStatus) {
  NJPenCommManPenConnectionStatusNone,
  NJPenCommManPenConnectionStatusScanStarted,
  NJPenCommManPenConnectionStatusConnected,
  NJPenCommManPenConnectionStatusDisconnected
  };
  -. The smartpen connection status will be notified with "NJPenCommManagerPenConnectionStatusChangeNotification" and it can be read with a status data as follows.

NSInteger penConnctionStatus = [[[notification userInfo] valueForKey:@"info"] integerValue];

- Bluetooth LE MTU size (iOS SDK2.0 only)
  BT LE MTU (Maximum Transmission Unit) size is the value which becomes decided by negotiation between the smartpen and the app in their connection.
- Offline data saving mode (iOS SDK2.0 only)
  It is a flag about whether or not offline data will be saved in the smartpen. (1: on, which means offline data will be saved, 0: off, offline data will not be saved.) It is returned by the smartpen when it is connected to the app. It will be triggered to be turned on automatically in the SDK while it is on the first connection procedure with the app after smartpens are mass-produced in a factory.

4.1.2 Input password

The default smartpen password is “0000”. But, if a smartpen doesn’t have the same password to the app, a correct password via a customized keypad should be input. After comparing a smartpen password, the smartpen connection process will be completed. Please refer to the 4.4.1 for a relevant method.

4.2 Stroke Data

If the application wants to get notified of the stroke in real time, the application requires to register NJPenCommParserStrokeHandler to NJPenCommManager by calling setPenCommParserStrokeHandler.
If there is no stroke handler registered when a stroke is delivered from Neo smartpen, penCommManager will notify with NJPenCommParserPageChangedNotification. This is to give an application a chance to register NJPenCommParserStrokeHandler to handle subsequent stroke data in real time. Usual application of handling real time stroke data is rendering the strokes on a view.

- (void) processStroke:(NSDictionary *)stroke
  -. It is called after a dictionary is set for stroke data and pen up/down status. A key 'type' is used to distinguish stroke and updown. A key 'node’ is used for a node information (x, y coordinate). And, a key ‘status’ has information about ‘up’ or ‘down’ for smartpen. The sample application uses it to render stroke.
- [pencommManager requestNewPageNotification]
  This is to let SDK informed there is a data for a new page when page canvas is closed, but, the same page is kept to be written.
- (void)findApplicableSymbolsX:(float)x andY:(float)y
  NPPUIInfo information(startX, startY, width, height, name, action, param) of symbol can be acquired by this method. Please refer to the example code of n2sample for more details.
- (void)notifyPageChanging
  This is to let the application informed there is a data for a new page. The application use this information to stop rendering subsequent stroke until getting notified with page opened notification.
- NJPage
  Class which includes the page information such as stroke data(NJStroke), paper size, notebook id, page number and so on. A NJPage object should be created before a page canvas is established. Please refer to NJPage.h for more details.
- NJStroke
  Class which includes the node information such as x, y coordinates.

Please refer to NJStroke.h for more details.

- NJNode
  Class for the node information such as x, y coordinates, pressure and time.
  Please refer to NJNode.h for more details.
- activeNoteBookId, activePageNumber
  -. activeNoteBookId is needed to know the ID of the note book which is being used and activePageNumber is needed to know the number of the page on which is being used
  -. activeNoteBookId and activePageNumber are necessary for a page canvas to be created and stroke data can be added on the page canvas while they are being written.
  -. They can be read by the below delegate method of the protocol "NJPenCommParserStrokeHandler".
  -. (void) activeNoteId:(int)noteId pageNum:(int)pageNumber sectionId:(int)section ownderId:(int)owner;
  : (int)noteId, (int)pageNumber, (int)section and (int)owner are note id, page page number, section id and owner id for the active page.
  But, for the first stroke as soon as a smartpen is connected, they can be read by the below delegate method of the protocol, "NJPenCommParserStartDelegate".
  -.(void) activeNoteIdForFirstStroke:(int)noteId pageNum:(int)pageNumber :sectionId:(int)section ownderId:(int)owner
  : note id, page number, section id and owner id for the page where the first stroke was added.
  -.(void) setPenCommNoteIDList :
  [pencommManager setAllNoteIdList] if you want to get all notes to be written on notes.
  [pencommManager setNoteIdListFromPList] if you want have notes list from "note_support_list.plist" applied and written on the note from the list.
  [pencommManager setNoteIdListSectionOwnerFromPList] if you want have all notes under section and owner ids from "note_support_list.plist" applied and written on the note. It means even though some note ids are not in the property list, but their section ids and owner ids are in the list, they can be written.
  [[NPPaperManager sharedInstance] reqAddUsingNote:notebookId section:sectionId owner:ownerId] It should be performed before 'set note id list' if you want to add notebook to be allowed to be used. Then, the note id which is added by this method will be sent to a smartpen with notebook ids from 'note_support_list.plist'
- (UInt32)setPenColor
  This is a delegate method under NJPenCommParserStrokeHandler protocol.
  Strokes color on canvas can be set through this method. Please refer to a new menu "Change canvas color" in the n2sample code for more details.

- (void) penCommMsg:(NSDictionary *)msg
  -. It is called when pen NDAC error occurs.
  This is a delegate method under NJPenCommParserStrokeHandler protocol.
  You can refer to what should be read in the sample code.

4.3 Rendering example

4.3.1 Simple rendering

The sample application draw lines to draw strokes in real time. It is implemented in NJPageCanvasController.m. Whenever processStroke is called it draw line with the dot received. When notebook background is needed to be displayed, both drawBG and opaque should be YES. When it is not, both options should be NO.

- (UIImage *) drawStroke: (NJStroke *)stroke withImage:(UIImage *)image size:(CGRect)bounds scale:(float)scale offsetX:(float)offset_x offsetY:(float)offset_y drawBG:(BOOL)drawBG opaque: (BOOL)opaque

The following method should be performed when you want to draw all strokes on a page at one time.

- (UIImage *) drawPageWithImage:(UIImage *)image size:(CGRect)bounds drawBG:(BOOL)drawBG opaque: (BOOL)opaque

4.3.2 Bezierpath rendering

If the application receives pen up, it means there is a complete stroke received. With the stroke data the sample application renders a stroke using bezierpath.
The sample application catches pen up state in processStroke and update a view with this rendering.

- (void)renderWithScale:(CGFloat)scale

4.4 Pen password

4.4.1 Setting NJPenPasswordDelegate and implementing a callback method

- Setting the delegate

```
[[NJPenCommManager sharedInstance] setPenPasswordDelegate:self];
```

- Callback method

```
- (void) penPasswordRequest:(PenPasswordRequestStruct *)request;
```

: retry count and reset count which were transmitted can be read from request.

4.4.2 Pen password input

```
[[NJPenCommManager sharedInstance] setBTComparePassword:password];
```

4.4.3 Pen password change

```
[[NJPenCommManager sharedInstance] changePasswordFrom:self.savedPin
To:self.currentPin];
```

4.4.4 Notification

If a password input is correct, the notification NJPenCommManagerPenConnectionStatusChangeNotification will be sent.
And if a password change is succeeded, the notification NJPenCommParserPenPasswordSutupSuccess will be sent.

4.5 Setting

4.5.1 Auto power on and sound

[[NJPenCommManager sharedInstance] setPenStateAutoPower:pAutoPwer Sound:pSound];

4.5.2 Shutdown Timer

[[NJPenCommManager sharedInstance] setPenStateWithAutoPwrOffTime:autoPwrOff];

4.5.3 Pen sensor pressure tuning

[[NJPenCommManager sharedInstance] setPenStateWithPenPressure:penPressure];

4.6 Offline Sync

In case of F120/50/D100, offline sync mode will be turned off when it is mass-produced. Thus, when it is connected to the app for the first time, this mode will be turned on in pen SDK while connection procedure.

4.6.1 Setting NJOffineDataDelegate and requesting Offline Sync file list

- The method by which requestOfflineFileList is performed.

[[NJPenCommManager sharedInstance] setOfflineDataDelegate:self];

- callback method for offline data file list. Note list can be acquired from noteListDic.

(void) offlineDataDidReceiveNoteList:(NSDictionary *)noteListDic

- callback method for offline data file note list count of a section and an owner ID

(void) offlineDataDidReceiveNoteListCount:(int)noteCount
ForSectionOwnerId:(UInt32)sectionOwnerId

4.6.2 Offline Sync file data request and receiving stroke data from smartpen

- requesting offline sync file data

(BOOL) requestOfflineDataWithOwnerId:(UInt32)ownerId noteId:(UInt32)noteId;

- callback method for offline sync data status while the data is being transmitted

(void) offlineDataReceiveStatus:(OFFLINE_DATA_STATUS)status percent:(float)percent

- receiving stroke data from a smartpen

(BOOL) parseOfflinePenData:(NSData *)penData
: penData is raw data which is given from a smartpen

(void) parseOfflineDots:(NSData *)penData startAt:(int)position
withFileHeader:(OffLineDataFileHeaderStruct *)pFileHeader
andStrokeHeader:(OffLineDataStrokeHeaderStruct *)pStrokeHeader

: Please refer to NeoPenService.h for the structures for “OffLineDataFileHeaderStruct” and “OffLineDataStrokeHeaderStruct”.
In addition, you can get section ID from nOwnerId which consists of section ID and Owner ID. For example, if nOwnerId is 50331675 and is converted to hex from decimal, it will be 0x300001B. The MSB 8bits is section ID (0x3) and the LSB 24bits is owner ID (0x1B, decimal:27).

(BOOL) parseSDK2OfflinePenData:(NSData *)penData
AndOfflineDataHeader:(OffLineData2HeaderStruct*)offlineDataHeader
: It should be used for SDK2.0. penData is raw data which is given from a smartpen. Please refer to NeoPenService.h for the structures for “OffLineData2HeaderStruct”

(void) parseSDK2OfflineDots:(NSData *)penData startAt:(int)position
withOfflineDataHeader:(OffLineData2HeaderStruct *)pFileHeader
andStrokeHeader:(OffLineData2StrokeHeaderStruct *)pStrokeHeader
: It should be used for SDK2.0. Please refer to NeoPenService.h for the structures for “OffLineData2HeaderStruct” and “OffLineData2StrokeHeaderStruct”.

- receiving a path to offline raw file before parsed.

(void) offlineDataPathBeforeParsed:(NSString *)path
: The offline raw file is the same as a smartpen has. If you want to save it in your app locally, you can use this method. However, it should be saved as soon as it is transmitted from the smartpen, because an offline file gets to be replaced by a next one in the same path.

4.7 Update pen firmware

Please select “Pen Firmware Update” from the menu to proceed updating pen firmware.

4.7.1 Setting NJFWUpdateDelegate

[[NJPenCommManager sharedInstance] setFWUpdateDelegate:self];

4.7.2 Sending pen firmware file to pen

[[NJPenCommManager sharedInstance] sendUpdateFileInfoAtUrlToPen:filePath];

4.7.2 Suspending sending pen firmware file

- canceFWUpdate flag should be set into YES if stopping sending pen firmware needs to be done.

[NJPenCommManager sharedInstance].cancelFWUpdate = YES;
If you click the back button while firmware update is being proceeded, it will stop with ‘cancelFWUpdate = YES’. Also, if you proceed updating firmware through ‘Pen Firmware Update’, it will send the firmware file to the pen from where you stopped in the previous update.

4.7.3 Read a firmware version

- [[NJPenCommManager sharedInstance] getFWVersion]
  Return a Firmware version string for a pen connected

4.7.4 Firmware update procedure

- Please refer to “NJFWUpdateViewController.m” of n2sample application for how firmware update is implemented.

1. delegate setting as follows when a viewcontroller for firmware update starts.
   =>[[NJPenCommManager sharedInstance] setFWUpdateDelegate:self];
2. read a version number and a location string from the json file of the following path
   SDK1.0 : http://one.neolab.kr/resource/fw/f1xx_firmware.json
   SDK2.0 : http://one.neolab.kr/resource/fw20/protocol2.0_firmware.json (F120)
   http://one.neolab.kr/resource/fw20/protocol2.0_firmware_f50.json (F50)
3. download a firmware version from the following server path.
   SDK1.0 : http://one.neolab.kr/resource/fw/ + location (from 2)
   SDK2.0 : http://one.neolab.kr/resource/fw20/ + location (from 2)
4. send the firmware version file downloaded from 2 to a N2 pen via the following API.
   =>[[NJPenCommManager sharedInstance] sendUpdateFileInfoAtUrlToPen:filePath]
5. you can get to know how much the firmware file transmits to the pen from the app via the following method.
   =>(void)fwUpdateDataReceiveStatus:(FW_UPDATE_DATA_STATUS)status percent:(float)percent
6. The blue led of the pen blinks (it means its firmware is being updated) when the pen resets by pressing a power button, if the firmware file transmits 100% successfully.

4.8 Pen status

4.8.1 Setting NJPenStatusDelegate and sending setPenState

- setting the delegate

[[NJPenCommManager sharedInstance] setPenStatusDelegate:self];

- sending pen state

[[NJPenCommManager sharedInstance] setPenState];

4.8.2 callback method

(void) penStatusData:(PenStateStruct *)penStatus

4.9 Transferable note Id

4.9.1 Setting transferable note id

[[NJPenCommManager sharedInstance] setNoteIdListFromPList];

4.10 Pen tip led color

4.10.1 Setting pen tip color led

[[NJPenCommManager sharedInstance] setPenStateWithRGB:penColor];

4.11 BT list for discovered peripherals

It returns peripherals array and uuid array discovered during BT peripherals scan.

4.11.1 Setting NJPenCommManagerNewPeripheral

- setting the delegate
  [[NJPenCommManager sharedInstance] setHandleNewPeripheral:self];
- BT scan command to have peripherals array and uuid array returned. Timer should be set after performing this command
  [[NJPenCommManager sharedInstance] btStartForPeripheralsList];
- The arrays should be read after timer expiry. Discovered peripherals will be collected during the time set by timer. You can check it with sample application menu “BT List”.
  [[NJPenCommManager sharedInstance] discoveredPeripherals];
- Command for connection with a pen which is selected
  [[NJPenCommManager sharedInstance] connectPeripheralAt: (NSInteger)index];

4.11.2 callback method

(void) connectionResult:(BOOL)success
: It returns connection result for “connectPeripheralAt” method.

4.12 Paper UI for email

An email client view will be presented, if you mark on an email icon of a note.

4.12.1 Setting NJPenCommParserCommandHandler

- setting the delegate

[[NJPenCommManager sharedInstance] setPenCommParserCommandHandler:self];

4.12.2 callback method

(void) sendEmailWithPdf
This delegate method will be called if an email icon is marked on a note.

(void) findApplicableSymbols:(NSString *)param action:(NSString *)action andName:(NSString *)name
This delegate method will provide the information for param, action and name if a specific area of a note is marked.

4.12.2 Generating ‘note_pui_info.plist’ (deprecated)

Please refer to the chapter 5 for how to set up the plist file.

4.13 Notification when a pen has already been connected by other apps

App will be notified with a delegate callback method, if a pen has already been connected by the one of other apps installed.

4.13.1 Setting NJPenCommParserCommandHandler

- setting the delegate

[[NJPenCommManager sharedInstance] setPenCommParserCommandHandler:self];

4.13.2 callback method

(void) penConnectedByOtherApp:(BOOL)penConnected
Boolean penConnected will be YES, if a pen has already been occupied by others.

If it is YES, your app will be able to notify with a pop-up message, for instance, “Your pen has been connected by the one of other apps. Please disconnect it from the app and please try again”

4.14 Battery level and used memory space of a pen

The current battery level and the used memory space of the pen can be read with the following method. Please refer to ‘getPenBatteryLevelAndMemoryUsedSpace’ or n2sample app menu ‘Battery Level and Memory Space’ at n2sample/NJViewController.m/ for more details.

(void) getPenBattLevelAndMemoryUsedSize:(void (^)(unsigned char remainedBattery, unsigned char usedMemory))completionBlock

4.15 NPPaperManager

It is designed to be used as singleton for notebook and page information from nproj file. The best practice to get instance of the Class looks like below.
[NPPaperManager sharedInstance];

(BOOL)installNotebookInfoForKeyName:(NSString *)keyName zipFilePath:(NSURL *)zipFilePath deleteExisting:(BOOL)deleteExisting;

This method is used to parse ‘nproj files’ from notebook zip files under ‘books_2016_02’ folder and save the parsed information into memory.

(NSString *) keyNameForNotebookId:(NSUInteger)notebookId section:(NSUInteger)section owner:(NSUInteger)owner

You can get a key name which is a combination of notebook ID, section and owner ID.

(void)reqAddUsingNote:(NSUInteger)notebookId section:(NSUInteger)sectionId owner:(NSUInteger)ownerId
This method is used to add notebook ID to note_support_list which should be transmitted to a pen for transferable note ID list configuration.

(UIImage *) getDefaultBackGroundImageForPageNum:(NSUInteger)pageNum NotebookId:(NSUInteger)notebookId section:(NSUInteger)section owner:(NSUInteger)owner
This method returns background image installed with zip file from book_2016_02 resource folder.

~~(NPPaperInfo *) getPaperInfoForNotebookId:(NSUInteger)notebookId pageNum:(NSUInteger)pageNum section:(NSUInteger)section owner:(NSUInteger)owner~~

You can get page information. You should input ‘notebook ID, page number, section ID and owner ID’ of the page of the notebook you would like to acquire. (deprecated)

You should use the following method to get page information instead of the above method. However, the following method has been implemented at NJNotebookPaperInfo class which is also designed to be used as a singleton for paper information from property list. The best practice to get

instance of the Class looks like [NJNotebookPaperInfo sharedInstance]. Please refer to n2sample codes for the usage.

(NPPaperInfo *) getNotePaperInfoForNotebook:(int)notebookId pageNum:(int)pageNum section:(int)sectionId owner:(int)ownerId

(NSString *) backgroundFileNameForSection:(int)sectionId owner:(UInt32)onwerId note:(UInt32)noteId

This method returns background image file string which is defined at note_paper_info.plist. It is also used with [NJNotebookPaperInfo sharedInstance].

(BOOL)section:(NSUInteger *)section owner:(NSUInteger *)owner fromNotebookId:(NSUInteger)notebookId

Class method. You can get section ID and owner ID for a notebook you would like to know. (deprecated) You need to create the method like this at your application. For example, if you are going to use a specific section ID and owner ID for your notebook, you should get this customized method to return the section ID and owner ID. Please refer to the same name class method at NJPage.m and the usage from n2sample.

5. Property list files creation

5.1 note_support_list.plist

This is used for setting transferable note IDs when a pen connection is initialized and established. A N2 pen can't be written on the notes which IDs are not included in the list. Please add note IDs which you want to use.

1. Go to n2sample/Resources/note_support_list.plist
   Add a new note id to the end line of section 3, owner 27 if its section id is 3 and owner id is 27.

5.2 note_paper_info.plist

A new note can be added to the property list as the following steps.

1. Go to n2sample/Resource/note_paper_info.plist
   Name "sectionId_ownerId_0NoteId" and add it to the plist.
   Add the number of pages, width, height, startX, startY, startPageNumber and bgFileName of the new note.

2. Add the new note background pdf file to n2sample/Resources/background directory.
   Add the new note cover image to n2sample/Resources and should get the cover image for the new note id in your app.

5.3 note_pui_info.plist (deprecated)

It can be edited with the following steps.

1. Go to n2sample/Resource/note_pui_info.plist
2. Name "note ID" and add it to the plist.
3. Add page number, startX, startY, width, height of an email icon on a page.

6. Nproj file parsing

You can add zip files which includes notebook nproj files, cover image and background image files as follows.

1. Add note_0xxx.zip file to 'books_2016_02' directory
    1) If there is one nproj file for one notebook, nproj name should be 'note_0noteId.nproj' (ex: note_0601.nproj for 601 notebook). Also, pdf and cover image file should be named same as nproj file. In the sample app (NISDK framework version 2.6), pdf file doesn't work, but jpg file works for notebook background file.
    2) If there are nproj files more than one for one notebook, nproj name should be 'sectionId_ownerId_noteId_order.nproj' (ex: 3_27_625_0.nproj, 3_27_625_1.nproj etc). Also, background jpg image files and cover image should be named same as nproj file. Furthermore, the extension of background jpg and png image file should be 'jpg' and 'png' (ex: 3_27_625_0.jpg, 3_27_625_1.jpg etc, 3_27_625_0.png, 3_27_625.1.png etc)
2. In the sample app (NISDK framework version 2.6), note_support_list.plist is needed for note list which is allowed to be written by N2 pen. Please refer to 5.1.

7. Sample application

7.1 Pen connection and offline sync test

You can test the sample application as follows.

1. There is a menu button on the sample application. If it is pressed, you can see a menu list.
2. Select "register" menu to register the pen by pressing its power button for 3 seconds (you can see a blue led is blinking if the power button is pressed for 3 seconds). If it is registered and connected successfully, you can see a white led on from the pen. This menu will be changed into "connect" if the pen is registered.
3. The menu list should be performed after the pen is connected with the sample app.

Also, You can test offline sync (data transfer from a pen) with the sample application as follows.

1. Turn on a pen by pressing a power button and write on a note with the pen.
2. There is a menu button on the sample application. If it is pressed, you can see a menu list.
3. Select "connect" menu to connect the pen.
4. Select "OfflineData list" to see the note list where stroke data was added and they were saved in the pen at the above number 1 step.
5. Select "Show OfflineData" to see the page which was offline-synchronized into the sample application.

7.2 BT ID test

You can apply BT IDs which you want to register for pen connection. If you set BT ID list before pen registration, only pen which BT ID is in the list will be registered to the app and it will be connected with the registered pen. You can test it as follows.

1. Select “BT ID” menu on the bottom of the menu list to set BT ID list. (It should be selected before selecting “Register”. “Register” menu will be displayed if you remove and install the sample app)
2. Select “Register” menu on the menu list.

(void) setBTIDForPenConnection:(NSArray *)btIDList

It should be performed before pen registration as bellows. Please refer to the sample app for more details.

//Please edit the following list you would like to allow to register to your app.

NSArray *btIDList = [NSArray arrayWithObjects:@"NWP-F50", @"NWP-F110", nil];

[[NJPenCommManager sharedInstance] setBTIDForPenConnection:btIDList];

However, pen registration with BT ID functionality will be disabled if SUPPORT_PEN_LOCALSUBNAME is undefined. If you want to register a pen with BT ID functionality via NISDK, SUPPORT_PEN_LOCALSUBNAME feature should be redefined in NISDK.

8. API index and description for header files

8.1 NJPenCommManager.h

@protocol NJPenCommManagerNewPeripheral
@optional
- (void) connectionResult:(BOOL)success;
@end
: Please refer to section 4.11 in page 14.

@protocol NJOfflineDataDelegate <NSObject>
- (void) offlineDataDidReceiveNoteList:(NSDictionary *)noteListDictionary;
- (BOOL) parseOfflinePenData:(NSData *)penData;
- (BOOL) parseSDK2OfflinePenData:(NSData *)penData
AndOfflineDataHeader:(OffLineData2HeaderStruct*)offlineDataHeader;
@optional
- (void) offlineDataReceiveStatus:(OFFLINE_DATA_STATUS)status percent:(float)percent;
- (void) offlineDataDidReceiveNoteListCount:(int)noteCount
ForSectionOwnerId:(UInt32)sectionOwnerId;

- (void) offlineDataPathBeforeParsed:(NSString *)path;
@end
: Please refer to section 4.6 in page 11.

@protocol NJPenCalibrationDelegate <NSObject>

```
@optional
- (void) calibrationResult:(BOOL)result;
@end
```
: It is not used

```
@protocol NJFWUpdateDelegate <NSObject>
@optional
- (void) fwUpdateDataReceiveStatus:(FW_UPDATE_DATA_STATUS)status percent:(float)percent;
@end
```
: Please refer to section 4.7 in page 13.

```
@protocol NJPenStatusDelegate <NSObject>
- (void) penStatusData:(PenStateStruct *)data;
@end
```
: Please refer to section 4.8 in page 13.

```
@protocol NJPenPasswordDelegate <NSObject>
- (void) penPasswordRequest:(PenPasswordRequestStruct *)data;
@end
```
: Please refer to section 4.4 in page 11.

```
@property (strong, nonatomic) NSMutableArray *discoveredPeripherals;
```
: Please refer to section 4.11 in page 14.

```
@property (nonatomic) BOOL cancelFWUpdate;
```
: Please refer to section 4.7 in page 13.

```
@property (nonatomic, readwrite) NJPenCommManPenConnectionStatus penConnectionStatus;
```
: Please refer to section 4.1 in page 8.

```
@property (nonatomic, readonly) BOOL isPenConnected;
```
: Please refer to section 4.1 in page 8.

```
@property (nonatomic, readwrite) BOOL hasPenRegistered;
```
: Please refer to section 4.1 in page 8.

```
@property (nonatomic, readonly) NSString *regUuid;
```
: Please refer to section 4.1 in page 8.

```
@property (nonatomic, readonly) NSString *penName;
```
: You can get this pen name which your pen has been transmitted as BT_ID, when it is registered.

```
@property (nonatomic) NSMutableArray *macArray;
```

: You can get mac list from pens around you during a specific time you set, when peripherals are scanned.

+ (NJPenCommManager *) sharedInstance;
: Please refer to section 2.1 in page 6.

- (void) setPenCommParserStrokeHandler:(id<NJPenCommParserStrokeHandler>)strokeHandler;

: Please refer to section 4.2 in page 9.

(void)setPenCommParserCommandHandler:(id<NJPenCommParserCommandHandler>)commandHandler;
: Please refer to section 4.12 in page 14.

- (void) setPenCommParserPasswordDelegate:(id<NJPenCommParserPasswordDelegate>)delegate;
: It is not used.

- (void) setPenCommParserStartDelegate:(id<NJPenCommParserStartDelegate>)delegate;
: Please refer to section 4.2 in page 9.

- (void) setOfflineDataDelegate:(id)offlineDataDelegate;
: Please refer to section 4.6 in page 11.

- (void) setPenCalibrationDelegate:(id)penCalibrationDelegate;
: It is not used.

- (void) setFWUpdateDelegate:(id)fwUpdateDelegate;
: Please refer to section 4.7 in page 12.

- (void) setPenStatusDelegate:(id)penStatusDelegate;
: Please refer to section 4.8 in page 13.

- (void) setPenPasswordDelegate:(id)penPasswordDelegate;
: Please refer to section 4.4 in page 11.

//Public API
- (void) btStart;
: Please refer to section 4.1 in page 8.

- (void) btStartForPeripheralsList;
: Please refer to section 4.11 in page 14.

- (void) btStop;
: Please refer to section 4.1 in page 8.

- (void) disConnect;
: Please refer to section 4.1 in page 8.

- (void) setPenState;
: Please refer to section 4.8 in page 13.

- (void)setNoteIdListFromPList;
: Please refer to section 4.2 in page 10.

- (void)setAllNoteIdList;
: Please refer to section 4.2 in page 10.

- (void)setNoteIdListSectionOwnerFromPList;
: Please refer to section 4.2 in page 10.

- (void)setPenStateWithRGB:(UInt32)color;
: Please refer to section 4.10 in page 14.

- (void)setPenThickness:(NSUInteger)thickness;
: To adjust pen thickness scale. thickness should be 1 ~ 3.

- (void)setPenStateWithPenPressure:(UInt16)penPressure;
: Please refer to section 4.5 in page 11.

- (void)setPenStateWithAutoPwrOffTime:(UInt16)autoPwrOff;
: Please refer to section 4.5 in page 11.

- (void)setPenStateAutoPower:(unsigned char)autoPower Sound:(unsigned char)sound;
: Please refer to section 4.5 in page 11.

- (void)setBTIDForPenConnection:(NSArray *)btIDList;

: Please refer to section 7.2 in page 18.

- (void)setPenStateWithTimeTick;

: To send device system time to a pen

- (unsigned char)getPenStateWithBatteryLevel;
: deprecated

- (unsigned char)getPenStateWithMemoryUsed;
: deprecated

- (NSString *)getFWVersion;
: Please refer to section 4.7 in page 13.

- (BOOL) requestOfflineDataWithOwnerId:(UInt32)ownerId noteId:(UInt32)noteId;
: Please refer to section 4.6 in page 12.

- (void) changePasswordFrom:(NSString *)curNumber To:(NSString *)pinNumber;
: Please refer to section 4.4 in page 11.

- (void) setBTComparePassword:(NSString *)pinNumber;
: Please refer to section 4.4 in page 11.

- (void) sendUpdateFileInfoAtUrlToPen:(NSURL *)fileUrl;
: Please refer to section 4.7 in page 13.

- (void) requestNewPageNotification;
: NJPenCommParserPageChangedNotification occurs when note or page id is changed. This method has the notification sent when the same page starts to be written again after a canvas view is closed.

- (void)resetPenRegistration;
: Please refer to section 4.1 in page 8.

- (void) connectPeripheralAt:(NSInteger)index;

: Please refer to section 4.11 in page 14.

- (void) getPenBattLevelAndMemoryUsedSize:(void (^)(unsigned char remainedBattery, unsigned char usedMemory))completionBlock;

: Please refer to section 4.14 in page 15.

8.2 NJPenCommParser.h

@protocol NJPenCommParserStrokeHandler <NSObject>

- (void) processStroke:(NSDictionary *)stroke;
- (void) activeNoteId:(int)noteId pageNum:(int)pageNumber sectionId:(int)section ownderId:(int)owner;
- (void) notifyPageChanging;
@optional
- (void) notifyDataUpdating:(BOOL)updating;
- (UInt32)setPenColor;
- (void) penCommMsg:(NSDictionary *)msg;
@end
: Please refer to section 4.2 in page 9.

@protocol NJPenCommParserCommandHandler <NSObject>
@optional
- (void) sendEmailWithPdf;
- (void) penConnectedByOtherApp:(BOOL)penConnected;
- (void) findApplicableSymbols:(NSString *)param action:(NSString *)action andName:(NSString *)name;
@end
: Please refer to section 4.12 and 4.13 in page 14 and page 15.

@protocol NJPenCommParserPasswordDelegate <NSObject>
- (void) performComparePassword:(PenPasswordRequestStruct *)request;
@end
: It is not used

@protocol NJPenCommParserStartDelegate <NSObject>
- (void) activeNoteIdForFirstStroke:(int)noteId pageNum:(int)pageNumber sectionId:(int)section ownderId:(int)owner;
- (void) setPenCommNoteIdList;
@end
: Please refer to section 4.2 in page 9.

8.3 NPPaperManager.h

- (NSArray *) notesSupported;
- (void)reqAddUsingNote:(NSUInteger)notebookId section:(NSUInteger)sectionId owner:(NSUInteger)ownerId;

- (NPPaperInfo *) getPaperInfoForNotebookId:(NSUInteger)notebookId pageNum:(NSUInteger)pageNum section:(NSUInteger)section owner:(NSUInteger)owner

- (BOOL)installNotebookInfoForKeyName:(NSString *)keyName zipFilePath:(NSURL *)zipFilePath deleteExisting:(BOOL)deleteExisting;

- (NSString *) keyNameForNotebookId:(NSUInteger)notebookId section:(NSUInteger)section owner:(NSUInteger)owner

: Please refer to section 4.2 in page 10 and page 15, 16.

8.3 NJNotebookPaperInfo.h

- (NPPaperInfo *) getNotePaperInfoForNotebook:(int)notebookId pageNum:(int)pageNum section:(int)sectionId owner:(int)ownerId;

: Please refer to section 4.15 in page 16.

- END -