iOS SDK

Specification Version 1.0.0

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	L.Park

	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	
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
08.FEB.2019	Document v2.3.5 release Add connectedPeripheral and rssiArray properties to make them available from pencommManager	L.Park
20.OCT.2020	NISDK3 v1.0.1 and Document v1.0.0 release NISDK3 is for Swift language.	S.B.Kim

1. Overview

This application note is for Neo smartpen iOS SDK3.0 (NISDK3 framework ver1.0.0).
Developing Neo smartpen applications 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, F121), Neo smartpen M1 (F50), Neo smartpen M1+ (F51), Neo smartpen Dimo (F30) should be v1.01 or above for the NISDK3 framework (ver. 1.0.0).

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 page.

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 device equipped with Bluetooth. This application note uses the 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 Requirements/Prerequisites

NISDK3 is for Swift4 and iOS v.10.0 or above.

2. SDK structure

The iOS NISDK3 sources are grouped 5 folders.

Name	Description
API	Pen Delegate
Common	Strings, definitions
Model	Structs, Enums
PenController	Bluetooth communication and protocol parser
Util	Utility functions

2.1 API

You can use protocol below here.
Use PenFinderDelegate to connect applications and Neo smartpen.
Use PenDelegate to communicate between applications and Neo smartpen.

2.2 PenController

Application uses PenFinder 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 an instance of the Class looks like below.

```
let penFinder = PenFinder.shared
```

PenFinder handles BTLE protocol related communications such as scan, connecting, disconnecting, etc.
PenCommParser handles protocol defined for communication between Neo smartpen and iPhone.

Please refer to the PenSearchViewController class. samples app for the example.
PenSearchViewCntroller conforms to PenFinderDelegate protocol.

If you click the button on the top of the View in the sample app, the app will commence scanning Neo smartpens for 10 seconds.
PenFinderDelegate will discover the pen and show the list of the Neo smartpens at the bottom sheet table.

If you select the table row,

```
penFinder.connectPeripheral(penList[row].peripheral)
```

will execute and set the pen at PenFinderDelegate - didConnect

2.3 pageData

The activity writing on a notebook is delivered to an iOS device as a group of coordinates 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 dot array to 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). RenderStrokeView represents a stroke.
While a user keeps writing on a page, there should be a lot of strokes delivered to an iOS device. PageStrokeView is a container of strokes written on the same page.
PageStrokeView and RenderStrokeView should be implemented on the application side for NISDK3 version 1.0.0 or above. Please refer to the NISDK3 sample 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), Neo smartpen M1+ (F51), Neo smartpen Dimo (F30) for NISDK3. The 3 easy steps are as follows:

- Step 1: Prerequisites
- Step 2: Prepare the NISDK3 for iOS
- 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 NISDK3 v1.0.0 for iOS supports iOS 10.x and higher.

The NISDK3 is written by Swift4.

3.2 Step 2: Prepare Neo smartpen SDK for iOS

The Neo smartpen N2 (F110, F120), Neo smartpen M1 (F50), Neo smartpen M1+ (F51), Neo smartpen Dimo (F30) SDK for iOS is a framework which is called "NISDK3".
NISDK3 is available through CocoaPods. To install it, simply add the following line to your Podfile:

```
pod 'NISDK3'
```

3.3 Step 3: Start coding

Import NISDK3 and CoreBluetooth(for use Bluetooth)
Start coding.

```
import NISDK3
import CoreBluetooth
```

4. Using iOS NISDK3

4.1 Controlling BTLE connection(connect/disconnect)

4.1.1 Connect/Disconnect

With an instance of PenFinder, you can scan the peripheral(Neo smartpen).

- PenFinder.shared.scan(10.0)
  Scan Neo smartpen for @parameter seconds.
- PenFinder.shared.disConnect(_ peripheral: CBPeripheral)
  Disconnect the peripheral(Neo smartpen).
- scanStop()
  Stop scanning Neo smartpen.

With an instance of PenFinderDelegate protocol,

- discoverPen(_ peripheral: CBPeripheral, _ pen: PenAdvertisementStruct, _ rssi: Int)
  If a pen is discovered, it gives peripheral info, pen info and rssi of the pen.
- scanStop()
  Stop scanning Neo smartpen.
- didConnect(_ pencontroller: PenController)
  Connected callBack.
- didFailToConnect(_ peripheral: CBPeripheral,_ error: Error?)
  Connecting fails with the peripheral.
- didDisconnect(_ central: CBCentralManager, _ peripheral: CBPeripheral?,_ error: Error?)
  Disconnected callBack.

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 4.6.1 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 PenDelegate - penData(_ sender: PenController, _ dot: Dot) to handle Stroke.

Applications can get Dot data using PenDelegate - penData.
Make Stroke using dots. And dots have 3 types.

- Down : Pen down - Pen starts writing. (Stroke start)
- Move: Pen move - Pen continuous writing.
- Up : Pen up - Pen finishes writing. (Stroke finish)

```
PenHelper.shared.dotDelegate = { [weak self] (dot)-> () in
        if dot.dotType == .Down{
        //down
        } else if dot.dotType == .Up{
        //up
        } else if dot.dotType == .Move{
        //move
        }
}
```

In the sample application MainViewController, you can see how to use dotDelegate.
Also, Dot data have page information.
Using these, you can check the page if changed.
Dot struct which includes such as x, y coordinate, force, tilt, pageInfo, penTipColor, and so on. Please refer to Dot struct for more details.

4.3 Note Data

With an instance of PenController,

- requestUsingNote(SectionOwnerNoteList list: [(UInt8,UInt32,UInt32?)])
  Request using specific note, @param[section, owner, note]
  If note is nil, using using all note
- requestUsingAllNote()
  Using all notes.
- PageInfo struct which includes the page information such as section, owner, note, page, eventcount and so on. When the dot is created, page info is setted. Please refer to PageInfo struct for more details.

4.4 Pen Data

When devices send messages to applications, penMessage(_ sender: PenController, _ msg: PenMessage) under PenDelegate protocol will be called.
Please refer to PenHelper in sample code, iIf you want more details.

4.5 Rendering example

4.5.1 Simple rendering

The sample application draws lines to draw strokes in real time. It is implemented in MainViewController. Whenever dotDelegate is called it draws a line with the dot received.
If an application wants to draw notebook background, use pageInfo in Dot struct.
Application gets notebook info using dot.pageInfo.
Sample app only offer 3 notebooks (601, 603, 655). If you want another notebook data, using .nproj file and parse it.

- pageStrokeView.addDot(_ dot: Dot)
  It draws real time strokes using dot data.
  Please refer to PageStrokeView in sample code, if you want more details.
- renderStrokeView.setStroke(_ dots: [Dot])
  It draws past strokes using dots.
  When dot.dotType is .Up, stroke is finished to draw. In that time, renderStokeView draws previous strokes.
  Please refer to RenderStrokeView in sample code, if you want more details.

4.5.2 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 uibezierpath.
The sample application catches pen up state in processStroke and updates a view with this rendering.

- drawStrokSimple(_ dots: [Dot])

4.6 Pen password

4.6.1 Setting PenAuthorizeDelegate and implementing a callback method

- Setting the delegate
  PenHelper.shared.penAutorizedDelegate
  It gives the pen connect state. If success returns true, else false.
  If false, pen requires password.
  If a pen fails to connect 10 times, pen data will initialize.

4.6.2 Pen password input

PenHelper.shared.pen?.requestComparePassword(_ pinNumber: String)

4.6.3 Pen password set

PenHelper.shared.pen?.requestSetPassword(_ pinNumber: String)

4.6.4 Pen password change

PenHelper.shared.pen?.requestChangePassword(from curNumber: String, to pinNumber: String)

4.7 Setting

If the application wants to change pen settings or get setting data, use penSettingDelegate.
Please refer to PenSettingViewController in sample code, if you want more details.

```
var penStatus: PenSettingStruct?

  override func viewDidLoad() {
    super.viewDidLoad()
    PenHelper.shared.penSettingDelegate = { [weak self] (status) in
      self?.penStatus = status
      DispatchQueue.main.async {
        self?.tableView.reloadData()
      }
    }
    PenHelper.shared.pen?.requestPenSettingInfo()
  }
```

4.7.1 Auto Power OnOff

PenHelper.shared.pen?.requestSetPenAutoPowerOn(_ onoff: OnOff)

4.7.2 Beep OnOff

PenHelper.shared.pen?.requestSetPenBeep(_ onoff: OnOff)

4.7.3 Shutdown Timer

PenHelper.shared.pen?.requestSetPenAutoPowerOffTime(_ minute: UInt16)

4.7.4 Pen Cap OnOff

PenHelper.shared.pen?.requestSetPenCapOff(_ onoff: OnOff)

4.7.5 Pen Hover OnOff

PenHelper.shared.pen?.requestSetPenHover(_ onoff: OnOff)

4.7.6 Pen Offline Data Save

PenHelper.shared.pen?.requestSetPenOfflineSave(_ onoff: OnOff)

4.8 Offline Sync

In case of F120/50/51/30, D100, offline sync mode will be turned off when it is mass-produced.
Thus, when application want receive offline data, call
PenHelper.shared.pen?.requestOfflineNoteList()

Please refer to PenOfflineNoteViewController in sample code, if you want more details.

4.8.1 Setting OffineDataDelegate and requesting Offline Sync file list

The method by which requestOfflineNoteList is performed.

- PenHelper.shared.offlinenoteDelegate

```
PenHelper.shared.offlinenoteDelegate = { [weak self] (noteinfo) -> () in
        self?.notelist = noteinfo.notes
        DispatchQueue.main.async {
          self?.tableView.reloadData()
        }
}
```

offlinenoteDelegate gives note info.

- PenHelper.shared.offlinepageDelegate

```
PenHelper.shared.offlinepageDelegate = { [weak self] (pageinfo) -> () in
        self?.pagelist = pageinfo
        DispatchQueue.main.async {
          self?.dataCheck = .page
          self?.tableView.reloadData()
        }
}
```

offlinenoteDelegate gives page info.

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

- requesting offline sync file data

requestOfflineData(_ section: UInt8, _ owner: UInt32, _ note: UInt32, _ pageList: [UInt32]?, _ deleteOnFinished: Bool)

Applications can use the offlinedataDelegate and parse data.

```
PenHelper.shared.offlinedataDelegate = { [weak self] (datainfo) -> () in
        self?.datalist = datainfo

        if self?.datalist?.strokeArray.count ?? 0 > 0 {

            DispatchQueue.main.async {
                let  vc = self?.navigationController?.viewControllers.filter({$0 is
MainViewController}).first
                PenHelper.shared.dotsDataDelegate!(self!.datalist!)
                self?.navigationController?.popToViewController(vc!, animated:
true)
            }
        }
}
```

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

var offlinestatusDelegate: ((_ percent: Float) -> ())?

- receiving stroke data from a smartpen

parseSDK2OfflinePenData(_ penData: [UInt8], _ offlineData: OffLineData)

penData is raw data which is given from a smartpen
Please refer to ResponseStruct.swift for the structures for ‘OfflineData’ and ‘OfflineStroke’.

4.9 Update pen firmware

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

4.9.1 Setting FWUpdateDelegate

PenHelper.shared.fwUpdateSuccessDelegate

4.9.2 Sending pen firmware file to pen

PenHelper.shared.pen?.UpdateFirmware(_ data: Data,_ deviceName: String,_ fwVersion : String)

- PenHelper.shared.pen?.setCancelFWUpdate()

If you click the “Cancel firmware update” button while firmware update is being proceeded, it will call cancelTask() and stop firmware update.

Please refer to “PenFWUpdateViewController” of NISDK3 sample application for how firmware update is implemented.

4.9.3 Read firmware version

- PenHelper.shared.pen?.requestPenVersionInfo()?.firmwareVersion
  Return a Firmware version string for a pen connected

4.9.4 Firmware update procedure

- Please refer to 'PenFWUpdateViewController' of NISDK3 sample application for how firmware update is implemented.

1. delegate setting as follows when a viewcontroller for firmware update starts.
   => PenHelper.shared.fwUpdateSuccessDelegate

2. read a version number and a location string from the json file of the following path : http://one.neolab.kr/resource/fw20/firmware_all_3.json

3. download a firmware version from the following server path.
   : 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.
   => PenHelper.shared.pen?.UpdateFirmware(_ data: Data,_ deviceName: String,_ fwVersion : String)

5. you can get to know how much the firmware file transmits to the pen from the app via the following method.
   => penFWUpgradePerDelegate: ((Float) -> ())?

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.

4.10 Pen status

- setting the delegate
  PenHelper.shared.penSettingDelegate

- request pen state
  PenHelper.shared.pen?.requestPenSettingInfo()

```
var penStatus: PenSettingStruct?

  override func viewDidLoad() {
    super.viewDidLoad()
    PenHelper.shared.penSettingDelegate = { [weak self] (status) in
      self?.penStatus = status
      DispatchQueue.main.async {
            self?.tableView.reloadData()
        }
    }
    PenHelper.shared.pen?.requestPenSettingInfo()

    let autoPowerOffTime = self.penStatus?.autoPwrOffTime
}
```

Please refer to “PenSettingViewController” of NISDK3 sample application.

4.11 Pen tip led color

4.11.1 Setting pen tip color led

requestSetPenLEDColor(_ color: LEDColor)

4.12 BT list for discovered peripherals

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

4.12.1 Setting PenFinderDelegate

- setting the delegate
  PenFinder.shared.delegate = self
- BT scan command to have peripherals array and uuid array returned. Timer should be set after performing this command
  PenFinder.shared.scan(_ second: CGFloat)

- The arrays should be read after timer expiry. Discovered peripherals will be collected during the time set by the timer. You can check it with the sample application if you click the left-top image button.

  PenFinderDelegate - discoverPen()

- Command for connection with a pen which is selected
  PenFinder.shared.connectPeripheral(_ peripheral: CBPeripheral)

4.12.2 callback method

PenFinderDelegate - didConnect
: It returns connection result for
‘PenFinder.shared.connectPeripheral(_ peripheral: CBPeripheral)’ method

Please refer to the sample app ‘PenSearchViewController’ class.
PenSearchViewCntroller conforms to PenFinderDelegate protocol.

4.13 Paper User Interface (PUI)

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

4.13.1 Setting SymbolActionProtocol

- setting the delegate
  ActionHelper.shared.delegate = self
- check if the symbol exists.
  ActionHelper.shared.symbolCheck(_ dots: [Dot)

4.13.2 callback method

```
func Event(symbol: SymbolData)
```

This delegate method in SymbolActionProtocol will be called if a symbol icon is marked on a note.

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.

```
penStatus?.battLevel
penStatus?.memoryUsed
```

Please refer to section 4.7 or sample app ‘PenSettingViewController’ for more details.

5. Nproj file parsing

We assume you have nproj file, note background image or pdf and cover image.

- Your nproj file name should be ‘note_{noteID}.nproj’ (ex. note_234.nproj for 234 notebook).
  Also, pdf and cover image file should be named the same as nproj file.

- 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 the 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)

- In the Sample app(NISDK3 framework version 1.0.0), have an example of .nproj file parsing and use background images.

Please refer to “MainViewController”, “NProjParser”, “ScaleHelper(for match dot to pixel)” of NISDK3 sample application for how to parse .nproj and use it.

6. Sample application

6.1 Pen connection and offline sync test

You can test the sample application as follows.

1. There is a menu image button(“pencil.slash”) top-left on the sample application. If it is pressed, you can see a menu list.
2. if you select the button, bottom sheet will appear and search for pens around you.
3. If pens discovered, table showing the list.
4. Select your pen 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. Image button will be changed into “pencil” if the pen is registered.
5. Top-right “*” image button will appear after the pen is connected with the sample app.
6. Select the button and use other options. (Change password, test pen offline data, etc.)

7. API index and description

7.1 PenFinder

- func initBluetooth()
  : start connection process
- func scan(_ second: CGFloat)
  : Please refer to section 4.1.1 on page 10.
- func scanStop()
  : Please refer to section 4.1.1 on page 10.

7.2 PenFinderDelegate

- func discoverPen(_ peripheral: CBPeripheral, _ pen: PenAdvertisementStruct, _ rssi: Int)
- func scanStop()
- func didConnect(_ pencontroller: PenController)
- func didFailToConnect(_ peripheral: CBPeripheral,_ error: Error?)
- func didDisconnect(_ central: CBCentralManager, _ peripheral: CBPeripheral?,_ error: Error?)

: Please refer to section4.1.1 on page 10, Sample app “PenSearchViewControoler” for more details.

7.3 PenController

- func setPenDelegate(_ delegate: PenDelegate)
  : Pen data Callback PenProtocol.

: Please refer to Sample app “PenHelper” for more details.
Almost PenController’s functions are wrapper functions of PenCommParser.

7.4 PenCommParser

7.4.1 Parse Data

- func parsePen2Data(_ data: [UInt8])
  : Parse pen data to packet.

- func parsePenDataPacket(_ packet: [UInt8])
  : Read and parse packet data.

: Please refer to Sample app “PenCommParser”, “PenController” for more details.

7.4.2 Pen Version and Setting Information

- func requestVersionInfo()
  : When connected with a pen, you should always ask first.
  Request pen version information.

- func requestPenSettingInfo()
  : Request pen setting information.

7.4.3 Pen Password

- func requestComparePasswordSDK2(_ pinNumber: String)
  : If the pen is not authorized, the app should unlock the pen with pin number.

- func requestSetPassword(_ pinNumber: String)
  : Request setting the pen lock with pin number.

- func requestChangePassword(_ curNumber: String, to pinNumber: String)
  : Request change password.

: Please refer to section 4.6 on page 12

7.4.4 Pen Setting

- func requestSetPenTime()
  : Send device system time to pen.

- func requestSetPenAutoPowerOffTime(_ minute: UInt16)
  : Setting pen auto power off time.

- func requestSetPenCapOff(_ onoff: OnOff)
  : If setted on, pen will turn on when pen cap off and pen will turn off when pen cap on.

- func requestSetPenAutoPowerOn(_ onoff: OnOff)
  : If setted on, the pen will turn on when the pen tip is pressed.

- func requestSetPenBeep(_ onoff: OnOff)
  : Setting pen beep on, off.

- func requestSetPenOfflineSave(_ onOff: OnOff)
  : Setting save offline data on, off.

- func requestSetPenHover(_ onoff: OnOff)
  : Setting pen hover mode on, off.
  Hover mode is the function of when pen is close to NCode (not pen tip down), you can receive dot data using hoverDelegate.
  Please refer to the Sample app "PenHelper - hoverData() function".

- func requestSetPenLEDColor(_ color: LEDColor)
  : Setting pen led color.

- func requestSetPenPressStep(_ step: UInt8)
  : Setting pen pressure sensitive.

: Please refer to section 4.7 on page 13

7.4.5 Note

- func requestUsingAllNote()
  : Request using all notes.

- func requestUsingNote(SectionOwnerNoteList list: [(UInt8,UInt32,UInt32?)])
  : @Param: Array(SectionId, OwnerId, NoteId)
  Request using specific notes.

: Please refer to section 4.3 on page 11

7.4.6 Offline Note Data

- func requestOfflineNoteList()
  : Request saved offline note list in pen.

- func requestOfflinePageList(_ section: UInt8, _ owner: UInt32, _ note: UInt32)
  : Request offline saved page list in pen.

- func requestOfflineData(_ section: UInt8, _ owner: UInt32, _ note: UInt32, _ pageList: [UInt32]?, _ deleteOnFinished: Bool)
  : Request offline data of note or page.

- func requestOfflineDataAck(_ packetId: UInt16, _ errCode: ErrorCode, _ transOption: REQ.OfflineAckTransOP)
  : Request ACK after offline data.

- func requestDeleteOfflineData(_ section: UInt8, _ owner: UInt32, _ noteList: [UInt32])
  : Request delete saved offline data in pen.

- func parseSDK2OfflinePenData(_ penData: [UInt8], _ offlineData: OffLineData)
  : Parse offline data.

: Please refer to Sample app "PenOfflineNoteViewController" for more details.

7.4.7 Pen Firmware update

- func updateFirmwareFirst(_ data: Data, _ deviceName: String, _ fwVersion: String,_ compress: Bool)
  : If you want to cancel, cancelFWUpdate variable to false.
  (PeonController - func setCancelFWUpdate())

- func updateFirmwareSecond(at fileOffset: UInt32, andStatus status: UInt8)

: Please refer to Sample app “PenFWUpdateViewController” for more details.

7.4.8 Pen Profile

- func createProfile(_ proFileName: String , _ password: [UInt8]) throws
- func deleteProfile (_ proFileName: String, _ password: [UInt8]) throws
- func getProfileInfo (_ proFileName: String) throws
- func writeProfileValue (_ proFileName: String, _ password: [UInt8] , _ data: [String : [UInt8]]) throws

: It is not used. Profile can save some datas in your pen.

8. Appendix

NISDK ver.3 is the Swift version of NISDK ver.2.
Use NISDK ver.2 if you are developing with Objective-C, link for SDK ver.2 is [Here].

8.1 Major Changes

NISDK 2.0	NISDK 3.0
Connect	
- (void) connectionResult:(BOOL)success;	PenHelper.shared.connectDelegate
- (void) btStart;	func initBluetooth()
- (void) btStartForPeripheralsList;	func scan(_ second: CGFloat)
- (void) btStop;	func scanStop()
- (void) disConnect;	func disConnect(_ peripheral: CBPeripheral)
- (void)resetPenRegistration;	deprecated
- (void) connectPeripheralAt:(NSInteger)index;	deprecated
- (void) setPenState;	Please refer to Section 4.7 on page 13.
Password	
- (void) penPasswordRequest:(PenPasswordRequestStruct *)data;	PenHelper.shared.pen?.requestSetPassword(_ pinNumber: String)
- (void) setPenPasswordDelegate:(id)penPasswordDelegate;	deprecated
- (void) changePasswordFrom:(NSString *)curNumber To:(NSString *)pinNumber;	PenHelper.shared.pen?.requestChangePassword(from curNumber: String, to pinNumber: String)

- (void) setBTComparePassword:(NSString *)pinNumber;	PenHelper.shared.pen?.requestComparePassword(_ pinNumber: String)
- (void) performComparePassword:(PenPasswordRequestStruct *)request;	deprecated
Status	
- (void) setPenStatusDelegate:(id)penStatusDelegate;	PenHelper.shared.penSettingDelegate
Parser	
(void)setPenCommParserCommandHandler:(id<NJPenCommParserCommandHandler>)commandHandler;	PenCommParser.parsePenDataPacket(_ packet: [UInt8]) : Please refer to section 7.4.1 on page 20
(void) setPenCommParserStartDelegate:(id<NJPenCommParserStartDelegate>)delegate;	PenCommParser.parsePen2Data(_ data: [UInt8]) : Please refer to section 7.4.1 on page 20
Stroke	
- (void) setPenCommParserStrokeHandler:(id<NJPenCommParserStrokeHandler>)strokeHandler;	PenHelper.shared.dotDelegate : Please refer to section 4.2 on page 10
- (void) processStroke:(NSDictionary *)stroke;	deprecated
- (void) activeNoteId:(int)noteId pageNum:(int)pageNumber sectionId:(int)section ownderId:(int)owner;	PenHelper.shared.dotDelegate - dot.pageInfo : Please refer to section 4.2 on page 10
- (void) notifyPageChanging;	deprecated
- (void) notifyDataUpdating:(BOOL)updating;	deprecated
- (UInt32)setPenColor;	deprecated
- (void) penCommMsg:(NSDictionary *)msg;	deprecated
Note	
- (void)setNoteIdListFromPList;	deprecated
- (void)setAllNoteIdList;	func requestUsingAllNote() : Please refer to section 4.3 on page 11
	func requestUsingNote(SectionOwnerNoteList list: [(UInt8,UInt32,UInt32?)]) : Please refer to section 4.3 on page 11
- (void)setNoteIdListSectionOwnerFromPList;	deprecated
- (void) requestNewPageNotification;	deprecated
- (void) activeNoteIdForFirstStroke:(int)noteId	deprecated

pageNum:(int)pageNumber sectionId:(int)section ownderId:(int)owner;	
- (void) setPenCommNoteIdList;	deprecated
- (void)reqAddUsingNote:(NSUInteger)notebookId section:(NSUInteger)sectionId owner:(NSUInteger)ownerId;	deprecated
Offline Data	
- (void) setOfflineDataDelegate:(id)offlineDataDelegate;	PenHelper.shared.offlinedataDelegate
- (void) offlineDataDidReceiveNoteList:(NSDictionary *)noteListDictionary;	PenHelper.shared.offlinenoteDelegate
- (BOOL) parseOfflinePenData:(NSData *)penData;	func parseSDK2OfflinePenData(_ penData: [UInt8], _ offlineData: OffLineData) : Please refer to section 7.4.6 on page 22
- (BOOL) parseSDK2OfflinePenData:(NSData *)penDataAndOfflineDataHeader:(OffLineData2HeaderStruct*)offlineDataHeader;	deprecated
- (void) offlineDataReceiveStatus:(OFFLINE_DATA_STATUS)status percent:(float)percent;	deprecated
	func requestOfflineDataAck(_ packetId: UInt16, _ errCode: ErrorCode, _ transOption: REQ.OfflineAckTransOP) { let request = REQ.OfflineDataAck(packetId, errCode, transOption)
- (void) offlineDataDidReceiveNoteListCount:(int)noteCount ForSectionOwnerId:(UInt32)sectionOwnerId;	deprecated
- (void) offlineDataPathBeforeParsed:(NSString *)path;	deprecated
- (BOOL) requestOfflineDataWithOwnerId:(UInt32)ownerId noteId:(UInt32)noteId;	func requestOfflineData(_ section: UInt8, _ owner: UInt32, _ note: UInt32, _ pageList: [UInt32]?, _ deleteOnFinished: Bool) : Please refer to section 7.4.6 on page 22
Firmware update	
- (void) setFWUpdateDelegate:(id)fwUpdateDelegate;	PenHelper.shared.fwUpdateSuccessDelegate
- (void) fwUpdateDataReceiveStatus:(FW_UPDATE_DATA_STATUS)status percent:(float)percent;	PenHelper.shared.penFWUpgradePerDelegate

- (void) sendUpdateFileInfoAtUrlToPen:(NSURL *)fileUrl;	PenHelper.shared.pen?.UpdateFirmware(_ data: Data,_ deviceName: String,_ fwVersion : String)
Pen Status	
- (void) penStatusData:(PenStateStruct *)data;	PenHelper.shared.penSettingDelegate
Pen Settings	
- (void) setPenStateWithRGB:(UInt32)color;	func requestSetPenLEDColor(_ color: LEDColor)
- (void) setPenThickness:(NSUInteger)thickness;	deprecated
- (void) setPenStateWithPenPressure:(UInt16)penPressure;	deprecated
- (void) setPenStateWithAutoPwrOffTime:(UInt16)autoPwrOff;	func requestSetPenAutoPowerOffTime(_ minute: UInt16)
- (void)setPenStateAutoPower:(unsigned char)autoPower Sound:(unsigned char)sound;	func requestSetPenBeep(_ onoff: OnOff)
- (void) setBTIDForPenConnection:(NSArray *)btIDList;	deprecated
- (void) setPenStateWithTimeTick;	deprecated
- (NSString *)getFWVersion;	PenHelper.shared.pen?.requestPenVersionInfo()?.firmwareVersion
- (void) getPenBattLevelAndMemoryUsedSize:(void (^)(unsigned char remainedBattery, unsigned char usedMemory))completionBlock;	in penSettingDelegate, - status.battLevel - status.memoryUsed - etc. : Please refer to section 4.7 on page 13.
PUI	
- (void) sendEmailWithPdf;	deprecated
- (void) findApplicableSymbols:(NSString *)param action:(NSString *)action andName:(NSString *)name;	deprecated
	protocol SymbolActionProtocol - func Event(symbol: SymbolData) : Please refer to section 4.13 on page 18.
Etc.	
- (void) penConnectedByOtherApp:(BOOL)penConnected;	deprecated

- End of Document-