DTDevices

Generated by Doxygen 1.8.3.1

Thu Feb 28 2013 09:43:57

Contents

Chapter 1

Module Documentation

1.1 Printer SDK

Provides access to PP-60, DPP-250 and DPP-350 printers.

Modules

- Delegate Notifications

  Notifications sent by the sdk on various events - barcode scanned, magnetic card data, communication status, etc.
- General functions

  Functions to connect/disconnect, set delegate, print graphic and text.
- Page Mode Functions

  Functions to work with the printer's page mode.
- Barcode Reader Functions

  Functions for scanning barcodes and direct control of the barcode engine.
- Magnetic Stripe Reader Functions

  Functions to work with the printer's magenetic card reader.
- SmartCard Reader Functions

  Functions to work with the printer's smart card reader.
- Mifare Reader Functions

  Functions to work with the printer's mifare cards reader.
- Table Functions

  Functions to create, fill and print tables.
- Cryptographic & Security Functions

  Cryptographical functions - loading keys, magnetic card encryption and printer authentication.

Classes

- protocol <PrinterDelegate>

  Protocol describing various notifications that PrinterSDK can send. More...
- class Printer

  Provides access to printer functions. More...

Macros

- #define **PRINTER_NO_EXCEPTIONS**
- #define STRUCTURES_DEFINED

  Connection state.
- #define LOGO_NORMAL 0

  Prints the logo at 203x203 DPI.
- #define LOGO_DOUBLEWIDTH 1

  Prints the logo at 101x203 DPI.
- #define LOGO_DOUBLEHEIGHT 2

  Prints the logo at 203x101 DPI.
- #define LOGO_DWDH 3

  Prints the logo at 101x101 DPI.
- #define INFO_BATVOLT 0

  Returns printer's battery voltage.
- #define INFO_BATPERCENT 1

  Returns printer's battery in percent.
- #define INFO_TEMPC 2

  Returns printer's head temperature in Celsius.
- #define INFO_TEMPFR 3

  Returns printer's head temperature in Fahrenheit.
- #define INFO_PRINTERVERSION 4

  Returns printer's firmware version.
- #define INFO_PRINTERMODEL 5

  Returns printer's model, one of the PRINTER_∗ constants.
- #define INFO_PAPERWIDTH 6

  Returns printer's paper width in pixels (203DPI)
- #define INFO_PAGEHEIGHT 7

  Returns printer's virtual page height in pixels (203DPI)
- #define **PRINTER_CMP10** 0
- #define **PRINTER_DPP350** 1
- #define **PRINTER_DPP250** 2
- #define **PRINTER_PP60** 3
- #define **PRINTER_DPP450** 4
- #define **PRINTER_EP60** 5
- #define **PRINTER_BL112** 10
- #define **CHANNEL_PRN** 1
- #define **CHANNEL_SMARTCARD** 2
- #define **CHANNEL_GPRS** 5
- #define **CHANNEL_ENCMSR** 14
- #define **CHANNEL_MIFARE** 16
- #define **PRN_STAT_BATTERY_LOW** 8
- #define **PRN_STAT_OVERHEAT** 16
- #define **PRN_STAT_PAPER_OUT** 32
- #define **MF_STAT_OK** 0
- #define **MF_STAT_TIMEOUT** -1
- #define **MF_STAT_COLLISION** -2
- #define **MF_STAT_PARITY_ERROR** -3
- #define **MF_STAT_FRAMING_ERROR** -4
- #define **MF_STAT_CRC_ERROR** -5
- #define **MF_STAT_FIFO_OVERFLOW** -6
- #define **MF_STAT_EEPROM_ERROR** -7
- #define **MF_STAT_INVALID_KEY** -8

- #define **MF_STAT_UNKNOWN_ERROR** -9
- #define **MF_STAT_AUTH_ERROR** -10
- #define **MF_STAT_CODE_ERROR** -11
- #define **MF_STAT_BITCOUNT_ERROR** -12
- #define **MF_STAT_NOT_AUTH** -13
- #define **MF_STAT_VALUE_ERROR** -14
- #define **MF_OPERATION_INCREMENT** 0xC0
- #define **MF_OPERATION_DECREMENT** 0xC1
- #define **MF_OPERATION_RESTORE** 0xC2
- #define **SCERR_NONE** 0
- #define **SCERR_FAILED** -1
- #define **SCERR_FILE_NOT_FOUND** -2
- #define **SCERR_RECORD_NOT_FOUND** -3
- #define **SCERR_INVALID_LENGTH** -4
- #define **SCERR_NO_FILE_SELECTED** -5
- #define BAR_PRN_UPCA 0

  Prints UPC-A barcode.
- #define BAR_PRN_UPCE 1

  Prints UPC-E barcode.
- #define BAR_PRN_EAN13 2

  Prints EAN-13 barcode.
- #define BAR_PRN_EAN8 3

  Prints EAN-8 barcode.
- #define BAR_PRN_CODE39 4

  Prints CODE39 barcode.
- #define BAR_PRN_ITF 5

  Prints ITF barcode.
- #define BAR_PRN_CODABAR 6

  Prints CODABAR barcode.
- #define BAR_PRN_CODE93 7

  Prints CODE93 barcode.
- #define BAR_PRN_CODE128 8

  Prints CODE128 barcode.
- #define BAR_PRN_PDF417 9

  Prints 2D PDF-417 barcode.
- #define BAR_PRN_CODE128AUTO 10

  Prints CODE128 optimized barcode.
- #define BAR_PRN_EAN128AUTO 11

  Prints EAN128 optimized barcode.
- #define **BAR_TEXT_NONE** 0
- #define **BAR_TEXT_ABOVE** 1
- #define **BAR_TEXT_BELOW** 2
- #define **BAR_TEXT_BOTH** 3
- #define **BAR_BOOKLAND** 0x16
- #define **BAR_CODABAR** 0x02
- #define **BAR_CODE11** 0x0C
- #define **BAR_CODE32** 0x20
- #define **BAR_CODE128** 0x03
- #define **BAR_CODE39** 0x01
- #define **BAR_CODE39_FULLASCII** 0x13
- #define **BAR_CODE93** 0x07
- #define **BAR_COMPOSITE** 0x1D
- #define **BAR_COUPON** 0x17

- #define **BAR_D25** 0x04
- #define **BAR_DATAMATRIX** 0x1B
- #define **BAR_EAN_128** 0x0F
- #define **BAR_EAN_13** 0x0B
- #define **BAR_EAN_13_PLUS_2** 0x4B
- #define **BAR_EAN_13_PLUS_5** 0x8B
- #define **BAR_EAN_8** 0x0A
- #define **BAR_EAN_8_PLUS_2** 0x4A
- #define **BAR_EAN_8_PLUS_5** 0x8A
- #define **BAR_IATA** 0x05
- #define **BAR_ISBT_128** 0x19
- #define **BAR_ISBT_128_CONCATENETED** 0x21
- #define **BAR_ITF** 0x06
- #define **BAR_MACROPDF** 0x28
- #define **BAR_MSI** 0x0E
- #define **BAR_PDF_417** 0x11
- #define **BAR_BAR_POSTBAR_CANADA** 0x26
- #define **BAR_POSTNET_US** 0x1E
- #define **BAR_POSTAL_AUSTRALIA** 0x23
- #define **BAR_POSTAL_JAPAN** 0x22
- #define **BAR_POSTAL_UK** 0x27
- #define **BAR_QR_CODE** 0x1C
- #define **BAR_RSS_LIMITED** 0x31
- #define **BAR_RSS_14** 0x30
- #define **BAR_RSS_EXPANDED** 0x32
- #define **BAR_SIGNATURE** 0x24
- #define **BAR_TRIOPTICCODE39** 0x15
- #define **BAR_UPCA** 0x08
- #define **BAR_UPCA_PLUS_2** 0x48
- #define **BAR_UPCA_PLUS_5** 0x88
- #define **BAR_UPCE** 0x09
- #define **BAR_UPCE_PLUS_2** 0x49
- #define **BAR_UPCE_PLUS_5** 0x89
- #define **BAR_UPCE1** 0x10
- #define **BAR_UPCE1_PLUS_2** 0x50
- #define **BAR_UPCE1_PLUS_5** 0x90
- #define **RESET_PRINTSETTINGS** 1
- #define **RESET_FONTSETTINGS** 2
- #define **RESET_BARCODESETTINGS** 4
- #define **RESET_DONTSETPRINTER** 0x80
- #define **ALIGN_LEFT** 0
- #define **ALIGN_CENTER** 1
- #define **ALIGN_RIGHT** 2
- #define **ALIGN_JUSTIFY** 3
- #define **TEXT_WORDWRAP** 1
- #define **TEXT_ROTATE_0** 0
- #define **TEXT_ROTATE_90** 1
- #define **TEXT_ROTATE_180** 2
- #define **LINESPACE_DEFAULT** 0x22
- #define **BLACKMARK_TRESHOLD_DEFAULT** 0x68
- #define **TABLE_BORDERS_HORIZONTAL** 1
- #define **TABLE_BORDERS_VERTICAL** 2
- #define **TABLE_COLUMN_COMPACT** 4
- #define PAGE_HORIZONTAL_TOPLEFT 0

Horizontal printing, starting from the top-left, continuing to the right.

- #define PAGE_VERTICAL_BOTTOMLEFT 1

  Vertical printing, starting from bottom-left, going upwards, newline goes right.
- #define PAGE_HORIZONTAL_BOTTOMRIGHT 2

  Horizontal printing, starting from the bottom-right, continuing to the left.
- #define PAGE_VERTICAL_TOPRIGHT 3

  Vertical printing, starting from top-right, going downwards, newline goes left.
- #define **XCOLORS_DEFINED**
- #define **ALG_AES256** 0
- #define **ALG_EH_ECC** 1
- #define **ALG_EH_AES256** 2
- #define **ALG_EH_IDTECH** 3
- #define KEY_AUTHENTICATION 0x00

  Authentication key.
- #define KEY_ENCRYPTION 0x01

  Encryption key, if set magnetic card data will be encrypted.
- #define KEY_EH_AES256_LOADING 0x02

  Encrypted head key loading key.
- #define KEY_EH_TMK_AES 0x10

  Encrypted head TMK key.
- #define KEY_EH_DUKPT_MASTER 0x20

  Encrypted head DUKPT master key.
- #define **FINANCIALCARD_DEFINED**

Enumerations

- enum **CONNSTATES** { **CONN_DISCONNECTED** =0, **CONN_CONNECTING**, **CONN_CONNECTED**, **CONN_CONNECT_FAILED** }
- enum BLUETOOTH_FILTER { BLUETOOTH_FILTER_ALL =-1, BLUETOOTH_FILTER_PRINTERS =1, BLUETOOTH_FILTER_PINPADS =2, BLUETOOTH_FILTER_BARCODE_SCANNERS =4 }

  Filtering bluetooth devices to discover.
- enum **COLORS** { **COLOR_WHITE** =0, **COLOR_BLACK**, **COLOR_INVERT** }

1.1.1 Detailed Description

Provides access to PP-60, DPP-250 and DPP-350 printers. In order to use PrinterSDK in your program, several steps have to be performed:

- Include PrinterSDK.h and libdtdev.a in your project.
- Go to Frameworks and add ExternalAccessory framework
- Edit your program plist file, add new element and select "Supported external accessory protocols" from the list, then add two items to it - com.datecs.printer.escpos and com.datecs.iserial.communication

1.1.2 Class Documentation

1.1.2.1 protocol PrinterDelegate-p

Protocol describing various notifications that PrinterSDK can send.

Instance Methods

- (void) - prnConnectionState:

  Notifies about the current connection state.
- (void) - paperStatus:

  Notification sent when printer's paper sensor changes.
- (void) - barcodeData:type:

  Notification sent when barcode is successfuly read.
- (void) - magneticCardData:track2:track3:

  Notification sent when magnetic card is successfuly read.
- (void) - magneticCardEncryptedData:tracks:data:

  Notification sent when magnetic card is successfuly.
- (void) - bluetoothDiscoverComplete:

  Notification sent when bluetooth discovery finds new bluetooth device.
- (void) - bluetoothDeviceDiscovered:name:

  Notification sent when bluetooth discovery finds new bluetooth device.
- (void) - bluetoothPrintingSupported:

  Indicates if bluetooth printing is supported, which happens when iSerial B accessory is connected.

1.1.2.2 class Printer

Provides access to printer functions.

Inherits NSObject.

Instance Methods

- (void) - addDelegate:

  Allows unlimited delegates to be added to a single class instance.
- (void) - removeDelegate:

  Removes delegate, previously added with addDelegate:(id)newDelegate.
- (void) - connect

  Connects to the device.
- (BOOL) - connectWithStreams:outputStream:error:

  Tries to find and connect to a printer via communication streams.
- (void) - disconnect

  Disconnects from the device.
- (BOOL) - **isPresent**
- (bool) - flushCache:

  Forces data still in the sdk buffers to be sent directly to the printer.
- (BOOL) - waitPrintJob:error:

  Waits specified timeout for the printout to complete.
- (int) - getPrinterStatus:

  Retrieves current printer status.
- (bool) - selfTest:error:

  Prints selftest.
- (bool) - turnOff:

  Forces printer to turn off.
- (bool) - feedPaper:error:

  Feeds the paper X lines (1/203 of the inch) or as needed (different length based on the printer model) so it allows paper to be teared.

- (bool) - printBarcode:barcode:error:

  Prints barcode.
- (bool) - printLogo:error:

  Prints the stored logo.
- (bool) - setBarcodeSettings:height:hriPosition:align:error:

  Set various barcode parameters.
- (bool) - setDensity:error:

  Sets printer density level.
- (bool) - setLineSpace:error:

  Sets the line "height" in pixels If the characters are 16 pixelx high for example, setting the linespace to 20 will make the printer leave 4 blank lines before next line of text starts.
- (bool) - setLeftMargin:error:

  Sets left margin.
- (bool) - printText:usingEncoding:error:

  Prints text with specified font/styles.
- (bool) - printText:error:

  Prints text with specified font/styles.
- (bool) - printDelimiter:error:

  Prints the delimiter character at the whole width of the paper, adjusting itself to the paper width.
- (bool) - getInfo:data:error:

  Returns different information about printer status/settings.
- (NSString ∗) - getPrinterSerialNumber:

  Returns printer unique serial number.
- (bool) - getBlackMarkTreshold:error:

  Returns blackmark sensor treshold or UnsupportedOperationException if printer is not in blackmark mode.
- (bool) - setBlackMarkTreshold:error:

  Sets blackmark sensor treshold or UnsupportedOperationException if printer is not in blackmark mode.
- (bool) - calibrateBlackMark:error:

  Provides blackmark sensor calibration by scaning 200mm of paper for possible black marks and adjust the sensor treshold.
- (bool) - beep:

  Makes short beep on the printer.
- (bool) - loadLogo:align:error:

  Loads logo into printer's memory.
- (void) - printImage:

  Prints Bitmap object.
- (bool) - printImage:align:error:

  Prints Bitmap object using specified alignment.
- (bool) - pageIsSupported

  Returns TRUE if page mode is supported on the connected device.
- (bool) - pageStart:

  Creates a new virtual page using the maximum supported page height.
- (bool) - pagePrint:

  Prints the content of the virtual page.
- (bool) - pageEnd:

  Exits page mode.
- (bool) - pageSetWorkingArea:top:width:height:error:

  Sets a working area and orientation inside the virtual page.
- (bool) - pageSetWorkingArea:top:width:heigth:orientation:error:

  Sets a working area and orientation inside the virtual page.
- (bool) - pageFillRectangle:error:

Fills the current working area (or whole page if none is set) with the specified color.

- (bool) - pageFillRectangle:top:width:height:color:error:

  Fills a rectangle inside the current working area with specified color.
- (bool) - pageRectangleFrame:top:width:height:framewidth:color:error:

  Draws a rectangle frame inside the current working area with specified color.
- (NSString ∗) - barcodeType2Text:

  Helper function to return string name of barcode type.
- (NSString ∗) - scanBarcode:timeout:error:

  Scans barcode using the built-in barcode scanning engine.
- (NSArray ∗) - msReadCard:error:

  Reads magnetic stripe card.
- (NSDictionary ∗) - msProcessFinancialCard:track2:

  Helper function to parse financial card and extract the data - name, number, expiration date.
- (bool) - scInit:

  Initializes and powers on the smartcard reader.
- (bool) - scClose:

  Powers down the smartcard reader.
- (NSData ∗) - scReset:

  Resets the smartcard and returns Answer To Reset.
- (NSData ∗) - scAPDU:ins:p1:p2:data:maxrcvlen:error:

  Sends an APDU command to the smartcard.
- (NSString ∗) - mfIdent:

  Returns mifare engine identification.
- (bool) - mfInit:

  Initializes and powers on the mifare reader module.
- (bool) - mfClose:

  Powers down mifare reader module.
- (bool) - mfRequestCards:rq1:rq2:error:

  Scans for mifare cards in the area.
- (bool) - mfAnticollision:error:

  Returns scanned card serial number.
- (bool) - mfSelectCard:sack:error:

  Select the card to use.
- (bool) - mfAuthByKey:block:key:error:

  Authenticate card block with direct key data.
- (NSData ∗) - mfRead:error:

  Reads a 16 byte block of data.
- (bool) - mfWrite:data:error:

  Writes a 16 byte block of data.
- (bool) - mfValueOperation:src_block:dst_block:value:error:

  Performs increment/decrement/restore operations.
- (bool) - mfGetReaderSerial:error:

  Returns mifare reader serial number.
- (bool) - mfWriteValue:value:error:

  Writes a 4 byte value in the card.
- (bool) - mfLoadKey:key:error:

  Stores key securely inside the mifare reader.
- (bool) - mfAuthByLoadedKey:block:keyID:error:

  Authenticate block by using previously stored key.
- (bool) - tableIsSupported

  Checks if the currently connected printer supports tables.

- (bool) - tableCreate:error:

  Create a new table using custom flags.
- (bool) - tableCreate:

  Create a new table using default settings - both horizontal and vertical borders around it.
- (bool) - tableAddColumn:

  Adds a new column using default settings - 12x24 font, plain, vertical border between the cells, left aligning.
- (bool) - tableAddColumn:error:

  Adds a new column using default settings - plain text, vertical border between the cells, left aligning.
- (bool) - tableAddColumn:style:alignment:error:

  Adds a new column using custom font and vertical border between the cells.
- (bool) - tableAddColumn:style:alignment:flags:error:

  Adds a new column.
- (bool) - tableAddCell:error:

  Adds a new cell using the font size and style and aligning of the column that cell belongs to.
- (bool) - tableAddCell:font:error:

  Adds a new cell using the font style and aligning of the column that cell belongs to.
- (bool) - tableAddCell:font:style:error:

  Adds a new cell using custom font size and style and aligning of the column that cell belongs to.
- (bool) - tableAddCell:font:style:alignment:error:

  Adds a new cell using custom font size and style and aligning.
- (bool) - tableAddDelimiter:

  Adds aa horizontal black line to the entire row that separates it from the next.
- (bool) - tableSetRowHeight:error:

  Sets the row height that will be used by default for new cells added.
- (bool) - tablePrint:

  Prints current table or throws IllegalArgumentException if cell data cannot be fit into paper.
- (NSData ∗) - cryptoRawGenerateRandomData:

  Generates 16 byte block of random numbers, required for some of the other crypto functions.
- (bool) - cryptoRawSetKey:encryptedData:error:
- (bool) - cryptoSetKey:key:oldKey:error:

  Used to store AES256 keys into printer's internal memory.
- (NSData ∗) - cryptoRawAuthenticatePrinter:error:
- (bool) - cryptoAuthenticatePrinter:error:

Class Methods

- (id) + sharedDevice

  Creates and initializes new Printer class instance or returns already initalized one.

Properties

- id delegate

  Adds delegate to the class.
- NSMutableArray ∗ delegates

  Provides a list of currently registered delegates.
- int connstate

  Returns current connection state.
- NSString ∗ deviceName

  Returns connected device name.
- NSString ∗ deviceModel

  Returns connected device model.

- NSString ∗ firmwareRevision

  Returns connected device firmware version.
- NSString ∗ hardwareRevision

  Returns connected device hardware version.
- NSString ∗ serialNumber

  Returns connected device serial number.
- int sdkVersion

  SDK version number in format MAJOR∗100+MINOR, i.e.

1.1.2.2.1 Property Documentation

1.1.2.2.1.1 - (int) sdkVersion [read],[atomic],[assign]

SDK version number in format MAJOR∗100+MINOR, i.e.

version 1.15 will be returned as 115

1.1.3 Macro Definition Documentation

1.1.3.1 #define BAR_PRN_CODE128AUTO 10

Prints CODE128 optimized barcode.

Supported only on DPP-350 and DPP-250 printers, it makes the barcode lot smaller especially when numbers only are used

1.1.3.2 #define BAR_PRN_EAN128AUTO 11

Prints EAN128 optimized barcode.

Supported only on DPP-350 and DPP-250 printers, it makes the barcode lot smaller especially when numbers only are used

1.1.3.3 #define INFO_BATPERCENT 1

Returns printer's battery in percent.

Note

Due to the way battery discharge, this information is not 100% accurate.

1.1.3.4 #define PAGE_HORIZONTAL_BOTTOMRIGHT 2

Horizontal printing, starting from the bottom-right, continuing to the left.

Newline goes up

1.1.3.5 #define PAGE_HORIZONTAL_TOPLEFT 0

Horizontal printing, starting from the top-left, continuing to the right.

Newline goes down

1.1.4 Enumeration Type Documentation

1.1.4.1 enum BLUETOOTH_FILTER

Filtering bluetooth devices to discover.

Enumerator

BLUETOOTH_FILTER_ALL Include all supported devices (default)

BLUETOOTH_FILTER_PRINTERS Include supported printers.

BLUETOOTH_FILTER_PINPADS Include supported pinpads.

BLUETOOTH_FILTER_BARCODE_SCANNERS Include supported barcode scanners.

1.2 Delegate Notifications

Notifications sent by the sdk on various events - barcode scanned, magnetic card data, communication status, etc.

Functions

- (void) - <PrinterDelegate>::prnConnectionState:

  Notifies about the current connection state.
- (void) - <PrinterDelegate>::paperStatus:

  Notification sent when printer's paper sensor changes.
- (void) - <PrinterDelegate>::barcodeData:type:

  Notification sent when barcode is successfuly read.
- (void) - <PrinterDelegate>::magneticCardData:track2:track3:

  Notification sent when magnetic card is successfuly read.
- (void) - <PrinterDelegate>::magneticCardEncryptedData:tracks:data:

  Notification sent when magnetic card is successfuly.
- (void) - <PrinterDelegate>::bluetoothDiscoverComplete:

  Notification sent when bluetooth discovery finds new bluetooth device.
- (void) - <PrinterDelegate>::bluetoothDeviceDiscovered:name:

  Notification sent when bluetooth discovery finds new bluetooth device.
- (void) - <PrinterDelegate>::bluetoothPrintingSupported:

  Indicates if bluetooth printing is supported, which happens when iSerial B accessory is connected.

1.2.1 Detailed Description

Notifications sent by the sdk on various events - barcode scanned, magnetic card data, communication status, etc.

1.2.2 Function Documentation

1.2.2.1 - (void) barcodeData: (NSString ∗) *barcode* type:(int) *type*

Notification sent when barcode is successfuly read.

Parameters

barcode	- string containing barcode data
type	- barcode type, one of the BAR_∗ constants

1.2.2.2 - (void) bluetoothDeviceDiscovered: (NSString ∗) *btAddress* name:(NSString ∗) *btName*

Notification sent when bluetooth discovery finds new bluetooth device.

Parameters

btAddress	bluetooth address of the device
btName	bluetooth name of the device

1.2.2.3 - (void) bluetoothDiscoverComplete: (BOOL) *success*

Notification sent when bluetooth discovery finds new bluetooth device.

Parameters

success	true if the discovery complete successfully, even if it not resulted in any device found, false if there was an error communicating with the bluetooth module

1.2.2.4 - (void) bluetoothPrintingSupported: (BOOL) *supported*

Indicates if bluetooth printing is supported, which happens when iSerial B accessory is connected.

Parameters

supported	- TRUE if bluetooth printing is supported and bt∗ functions can be used

1.2.2.5 - (void) magneticCardData: (NSString ∗) *track1* track2:(NSString ∗) *track2* track3:(NSString ∗) *track3*

Notification sent when magnetic card is successfuly read.

Parameters

track1	- data contained in track 1 of the magnetic card or nil
track2	- data contained in track 2 of the magnetic card or nil
track3	- data contained in track 3 of the magnetic card or nil

1.2.2.6 - (void) magneticCardEncryptedData: (int) *encryption* tracks:(int) *tracks* data:(NSData ∗) *data*

Notification sent when magnetic card is successfuly.

The data is encrypted via the selected encryption algorithm.

After decryption, the result data will be as follows:

- Random data (4 bytes)
- Device identification text (16 ASCII characters, unused bytes are 0)
- Processed track data in the format: 0xF1 (track1 data), 0xF2 (track2 data) 0xF3 (track3 data). It is possible some of the tracks will be empty, then the identifier will not be present too, for example 0xF1 (track1 data) 0xF3 (track3 data)
- End of track data (byte 0x00)
- CRC16 (2 bytes) - the CRC is performed from the start of the encrypted block (the Random Data block) to the end of the track data (including the 0x00 byte). The data block is rounded to 16 bytes

In the more secure way, where the decryption key resides in a server only, the card read process will look something like:

- (User) swipes the card
- (iOS program) receives the data via magneticCardEncryptedData and sends to the server
- (iOS program)[optional] sends current printer serial number along with the data received from magneticCard-EncryptedData. This can be used for data origin verification
- (Server) decrypts the data, extracts all the information from the fields
- (Server)[optional] if the ipod program have sent the printer serial number before, the server compares the received serial number with the one that's inside the encrypted block
- (Server) checks if the card data is the correct one, i.e. all needed tracks are present, card is the same type as required, etc and sends back notification to the ipod program.

Parameters

encryption	encryption algorithm used, one of: 0 — AES 256
tracks	shows which tracks have been read and inside the encrypted array, bits 0-2 represents track 1-3
data	contains the encrypted card data

1.2.2.7 - (void) paperStatus: (BOOL) *present*

Notification sent when printer's paper sensor changes.

Parameters

present	TRUE if paper is present, FALSE if printer is out of paper or cover is open

1.2.2.8 - (void) prnConnectionState: (int) *state*

Notifies about the current connection state.

Parameters

state	- connection state, one of: CONN_DISCONNECTED — there is no connection to the printer and the sdk will not try to make one even if the device is attached CONN_CONNECTING — Printer is not currently connected, but the sdk is actively trying to CONN_CONNECTED — Printer is connected

1.3 General functions

Functions to connect/disconnect, set delegate, print graphic and text.

Functions

- (id) + Printer::sharedDevice

  Creates and initializes new Printer class instance or returns already initalized one.
- (void) - Printer::addDelegate:

  Allows unlimited delegates to be added to a single class instance.
- (void) - Printer::removeDelegate:

  Removes delegate, previously added with addDelegate:(id)newDelegate.
- (void) - Printer::connect

  Connects to the device.
- (BOOL) - Printer::connectWithStreams:outputStream:error:

  Tries to find and connect to a printer via communication streams.
- (void) - Printer::disconnect

  Disconnects from the device.
- (BOOL) - **Printer::isPresent**
- (bool) - Printer::flushCache:

  Forces data still in the sdk buffers to be sent directly to the printer.
- (BOOL) - Printer::waitPrintJob:error:

  Waits specified timeout for the printout to complete.
- (int) - Printer::getPrinterStatus:

  Retrieves current printer status.
- (bool) - Printer::selfTest:error:

  Prints selftest.
- (bool) - Printer::turnOff:

  Forces printer to turn off.
- (bool) - Printer::feedPaper:error:

  Feeds the paper X lines (1/203 of the inch) or as needed (different length based on the printer model) so it allows paper to be teared.
- (bool) - Printer::printBarcode:barcode:error:

  Prints barcode.
- (bool) - Printer::printLogo:error:

  Prints the stored logo.
- (bool) - Printer::setBarcodeSettings:height:hriPosition:align:error:

  Set various barcode parameters.
- (bool) - Printer::setDensity:error:

  Sets printer density level.
- (bool) - Printer::setLineSpace:error:

  Sets the line "height" in pixels If the characters are 16 pixelx high for example, setting the linespace to 20 will make the printer leave 4 blank lines before next line of text starts.
- (bool) - Printer::setLeftMargin:error:

  Sets left margin.
- (bool) - Printer::printText:usingEncoding:error:

  Prints text with specified font/styles.
- (bool) - Printer::printText:error:

  Prints text with specified font/styles.
- (bool) - Printer::printDelimiter:error:

  Prints the delimiter character at the whole width of the paper, adjusting itself to the paper width.

- (bool) - Printer::getInfo:data:error:

  Returns different information about printer status/settings.
- (NSString ∗) - Printer::getPrinterSerialNumber:

  Returns printer unique serial number.
- (bool) - Printer::getBlackMarkTreshold:error:

  Returns blackmark sensor treshold or UnsupportedOperationException if printer is not in blackmark mode.
- (bool) - Printer::setBlackMarkTreshold:error:

  Sets blackmark sensor treshold or UnsupportedOperationException if printer is not in blackmark mode.
- (bool) - Printer::calibrateBlackMark:error:

  Provides blackmark sensor calibration by scaning 200mm of paper for possible black marks and adjust the sensor treshold.
- (bool) - Printer::beep:

  Makes short beep on the printer.
- (bool) - Printer::loadLogo:align:error:

  Loads logo into printer's memory.
- (void) - Printer::printImage:

  Prints Bitmap object.
- (bool) - Printer::printImage:align:error:

  Prints Bitmap object using specified alignment.

1.3.1 Detailed Description

Functions to connect/disconnect, set delegate, print graphic and text.

1.3.2 Function Documentation

1.3.2.1 - (void) addDelegate: (id) *newDelegate*

Allows unlimited delegates to be added to a single class instance.

This is useful in the case of global class and every view can use addDelegate when the view is shown and removeDelegate when no longer needs to monitor events

Parameters

newDelegate	the delegate that will be notified of Printer events

1.3.2.2 - (bool) beep: (NSError ∗∗) *error*

Makes short beep on the printer.

Parameters

error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.3.2.3 - (bool) calibrateBlackMark: (int ∗) *treshold* error:(NSError ∗∗) *error*

Provides blackmark sensor calibration by scaning 200mm of paper for possible black marks and adjust the sensor treshold.

Make sure you have put the right paper before calling this function.

Returns

returns new trashold value for the scanned paper. The trashold is already stored in printer's flash memory so no additional set is needed.

Parameters

error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.3.2.4 - (void) connect

Connects to the device.

Connection status will be passed via the delegate. The sdk will manage direct connections automatically, for example PP-60 connecting to the iOS device will trigger connection event. If you plug and unplug iSerial cable, connected to DPP-250 or DPP-350 it will be automatically detected too, but connection and disconnections from the other end of the iSerial cable cannot be autodetected. Calling this function will force try to detect the printer. If you are using the DPP-250 and DPP-350 printers via iSerial cable, it might be good idea to provide connect/disconnect buttons in the program, if the user is expected to disconnect the printers from the iSerial cable at random time.

1.3.2.5 - (BOOL) connectWithStreams: (NSInputStream ∗) *inStream* outputStream:(NSOutputStream ∗) *outStream* error:(NSError ∗∗) *error*

Tries to find and connect to a printer via communication streams.

This differs from the normal connect function - it is synchronious function and does not continue to automatically connect in the background. The use for this function is to connect to the printer via bluetooth streams (from Linea for example)

Parameters

inStream	input stream (bluetooth/socket/etc)
outStream	output stream (bluetooth/socket/etc)
error	returns error information, you can pass nil if you don't want it

Returns

true if connection was successful and printer is ready to receive information, false otherwise

1.3.2.6 - (void) disconnect

Disconnects from the device.

Connection status will be passed via the delegate

1.3.2.7 - (bool) feedPaper: (int) *lines* error:(NSError ∗∗) *error*

Feeds the paper X lines (1/203 of the inch) or as needed (different length based on the printer model) so it allows paper to be teared.

Note

If blackmark mode is active, this function searches for blackmark. If the paper is not blackmark one or the mark can not be found in 360mm, the printer will put itself into out of paper state and will need LF button to be pushed to continue.

Parameters

lines	the number of lines (1/203 of the inch) to feed or 0 to automatically feed the paper as much as needed to tear the paper.
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.3.2.8 - (bool) flushCache: (NSError ∗∗) *error*

Forces data still in the sdk buffers to be sent directly to the printer.

Parameters

error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.3.2.9 - (bool) getBlackMarkTreshold: (int ∗) *treshold* error:(NSError ∗∗) *error*

Returns blackmark sensor treshold or UnsupportedOperationException if printer is not in blackmark mode.

This value tells the printer how dark a spot on the paper needs to be in order to be considered as blackmark.

Parameters

treshold	upon success stores the current blackmark treshold
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.3.2.10 - (bool) getInfo: (int) *infocmd* data:(int ∗) *data* error:(NSError ∗∗) *error*

Returns different information about printer status/settings.

Parameters

infocmd	information type requested, one of the INFO_∗ constants
data	upon success stores the answer from the imfo command
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.3.2.11 - (NSString ∗) getPrinterSerialNumber: (NSError ∗∗) *error*

Returns printer unique serial number.

Returns

serial number

Parameters

error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.3.2.12 - (int) getPrinterStatus: (NSError ∗∗) *error*

Retrieves current printer status.

This function is useful on printers having no automatic status notifications like DPP-250 and DPP-350.

Parameters

error	returns error information, you can pass nil if you don't want it

Returns

one or more of the PRN_STAT_∗ constants or -1 if function failed

1.3.2.13 - (bool) loadLogo: (UIImage ∗) *logo* align:(int) *align* error:(NSError ∗∗) *error*

Loads logo into printer's memory.

The logo is persistent and can be deleted only if battery is removed

Parameters

logo	logo bitmap data
align	logo alignment, one of the ALIGN_∗ constants
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.3.2.14 - (bool) printBarcode: (int) *bartype* barcode:(NSData ∗) *barcode* error:(NSError ∗∗) *error*

Prints barcode.

Parameters

bartype	Barcode type, one of the BAR_PRN_∗ constants
barcode	barcode data to be printed
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.3.2.15 - (bool) printDelimiter: (char) *delimchar* error:(NSError ∗∗) *error*

Prints the delimiter character at the whole width of the paper, adjusting itself to the paper width.

The character is printed with font 12x24

Parameters

delimiter	character to print
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.3.2.16 - (void) printImage: (UIImage ∗) *image*

Prints Bitmap object.

You can print color bitmaps, as they will be converted to black and white using error diffusion and dithering to achieve best results. On older devices this can take some time

Parameters

image	UIImage object
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.3.2.17 - (bool) printImage: (UIImage ∗) *image* align:(int) *align* error:(NSError ∗∗) *error*

Prints Bitmap object using specified alignment.

You can print color bitmaps, as they will be converted to black and white using error diffusion and dithering to achieve best results. On older devices this can take some time

Parameters

image	UIImage object
align	image alighment, one of the ALIGN_∗ constants
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.3.2.18 - (bool) printLogo: (int) *mode* error:(NSError **) *error*

Prints the stored logo.

You can upload log with logo function

Parameters

mode	logo mode, one of the LOGO_* constants
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.3.2.19 - (bool) printText: (NSString *) *textString* error:(NSError **) *error*

Prints text with specified font/styles.

This function can act as both simple plain text printing and quite complex printing using internal tags to format the text. The function uses the currently font size and style (or default ones) as well as the aligning, however it allows modifications of them inside the text. Any modification of the settings using the tags will be reverted when function completes execution. For example if you have default font selected before using printText and set bold font inside, it will be reverted to plain when function completes. The tags are control commands used to modify the text printing parameters. They are surrounded by {} brackets. A list of all control tags follows:

- {==} - reverts all settings to their defaults. It includes font size, style, aligning
- {=Fx} - selects font size. x ranges from 0 to 1 as follows:
- 0: FONT_9X16 (hieroglyph characters are using the same width as height, i.e. 16x16)
- 1: FONT_12X24 (hieroglyph characters are using the same width as height, i.e. 24x24)
- {=L} - left text aligning
- {=C} - center text aligning
- {=R} - right text aligning
- {=Rx} - text rotation as follows:
- 0: not rotated
- 1: rotated 90 degrees
- 2: rotated 180 degrees
- {+/-B} - sets or unsets bold font style
- {+/-I} - sets or unsets italic font style
- {+/-U} - sets or unsets underline font style
- {+/-V} - sets or unsets inverse font style
- {+/-W} - sets or unsets text word-wrapping
- {+/-DW} - sets or unsets doubled font width

- {+/-DH} - sets or unsets doubled font height

An example of using tags "{=C}Plain centered text\n{=L}Left centered\n{+B}...bold...{-B}{+I}or ITALIC"

Parameters

textString	the text to print
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.3.2.20 - (bool) printText: (NSString ∗) *textString* usingEncoding:(NSStringEncoding) *encoding* error:(NSError ∗∗) *error*

Prints text with specified font/styles.

This function can act as both simple plain text printing and quite complex printing using internal tags to format the text. The function uses the currently font size and style (or default ones) as well as the aligning, however it allows modifications of them inside the text. Any modification of the settings using the tags will be reverted when function completes execution. For example if you have default font selected before using printText and set bold font inside, it will be reverted to plain when function completes. The tags are control commands used to modify the text printing parameters. They are surrounded by {} brackets. A list of all control tags follows:

- {==} - reverts all settings to their defaults. It includes font size, style, aligning
- {=Fx} - selects font size. x ranges from 0 to 1 as follows:
- 0: FONT_9X16 (hieroglyph characters are using the same width as height, i.e. 16x16)
- 1: FONT_12X24 (hieroglyph characters are using the same width as height, i.e. 24x24)
- {=L} - left text aligning
- {=C} - center text aligning
- {=R} - right text aligning
- {=Rx} - text rotation as follows:
- 0: not rotated
- 1: rotated 90 degrees
- 2: rotated 180 degrees
- {+/-B} - sets or unsets bold font style
- {+/-I} - sets or unsets italic font style
- {+/-U} - sets or unsets underline font style
- {+/-V} - sets or unsets inverse font style
- {+/-W} - sets or unsets text word-wrapping
- {+/-DW} - sets or unsets doubled font width
- {+/-DH} - sets or unsets doubled font height

An example of using tags "{=C}Plain centered text\n{=L}Left centered\n{+B}...bold...{-B}{+I}or ITALIC"

Parameters

textString	the text to print
encoding	the encoding to use when converting the string to format suitable to the printer. Default encoding should be NSWindowsCP1252StringEncoding. Currently double-byte encodings like JIS are not supported.
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.3.2.21 - (void) removeDelegate: (id) *newDelegate*

Removes delegate, previously added with addDelegate:(id)newDelegate.

Parameters

newDelegate	the delegate that will be no longer be notified of printer events

1.3.2.22 - (bool) selfTest: (BOOL) *longtest* error:(NSError $**$) *error*

Prints selftest.

Parameters

longtest	TRUE if you want complete test with fonts and codepage, FALSE for short one
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.3.2.23 - (bool) setBarcodeSettings: (int) *scale* height:(int) *height* hriPosition:(int) *hriPosition* align:(int) *align* error:(NSError $**$) *error*

Set various barcode parameters.

Parameters

scale	width of each barcode column in pixels (1/203 of the inch) between 2 and 4, default is 3
height	barcode height in pixels between 1 and 255. Default is 77
hriPosition	barcode hri code position, one of the BAR_TEXT_$*$ constants
align	barcode aligning, one of the ALIGN_$*$ constants
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.3.2.24 - (bool) setBlackMarkTreshold: (int) *treshold* error:(NSError $**$) *error*

Sets blackmark sensor treshold or UnsupportedOperationException if printer is not in blackmark mode.

This value tells the printer how dark a spot on the paper needs to be in order to be considered as blackmark.

Parameters

treshold	value between 0x20 and 0xc0, default is 0x68

Exceptions

NSPortTimeoutException	if there is no connection to the printer

1.3.2.25 - (bool) setDensity: (int) *percent* error:(NSError **) *error*

Sets printer density level.

Parameters

percent	density level in percents (50%-200%)
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.3.2.26 - (bool) setLeftMargin: (int) *leftMargin* error:(NSError **) *error*

Sets left margin.

Parameters

margin	left margin in pixels. Default is 0
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.3.2.27 - (bool) setLineSpace: (int) *lineSpace* error:(NSError **) *error*

Sets the line "height" in pixels If the characters are 16 pixelx high for example, setting the linespace to 20 will make the printer leave 4 blank lines before next line of text starts.

You cannot make text lines overlap.

Parameters

lineSpace	linespace in pixels, or 0 for automatic calculation. Default is 0
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.3.2.28 + (id) sharedDevice

Creates and initializes new Printer class instance or returns already initalized one.

Use this function, if you want to access the class from different places

Returns

shared class instance

1.3.2.29 - (bool) turnOff: (NSError **) *error*

Forces printer to turn off.

Parameters

error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.3.2.30 - (BOOL) waitPrintJob: (NSTimeInterval) *timeout* error:(NSError **) *error*

Waits specified timeout for the printout to complete.

It is best to call this function with the complete timeout you are willing to wait, rather than calling it in a loop

Parameters

timeout	the timeout to wait for the job to finish
error	returns error information, you can pass nil if you don't want it

Returns

TRUE if printer have successfully finished printing and ready to accept new data, FALSE if communication problem or the printer is still busy

1.4 Page Mode Functions

Functions to work with the printer's page mode.

Functions

- (bool) - Printer::pageIsSupported

  Returns TRUE if page mode is supported on the connected device.
- (bool) - Printer::pageStart:

  Creates a new virtual page using the maximum supported page height.
- (bool) - Printer::pagePrint:

  Prints the content of the virtual page.
- (bool) - Printer::pageEnd:

  Exits page mode.
- (bool) - Printer::pageSetWorkingArea:top:width:height:error:

  Sets a working area and orientation inside the virtual page.
- (bool) - Printer::pageSetWorkingArea:top:width:heigth:orientation:error:

  Sets a working area and orientation inside the virtual page.
- (bool) - Printer::pageFillRectangle:error:

  Fills the current working area (or whole page if none is set) with the specified color.
- (bool) - Printer::pageFillRectangle:top:width:height:color:error:

  Fills a rectangle inside the current working area with specified color.
- (bool) - Printer::pageRectangleFrame:top:width:height:framewidth:color:error:

  Draws a rectangle frame inside the current working area with specified color.

1.4.1 Detailed Description

Functions to work with the printer's page mode. Page mode is a special operation mode, that allows you to define a virtual page and then draw inside text, graphics, barcodes and print it all at once. Page mode allows for extended positioning of the elements, rotation, inversion and basic graphics elements.

1.4.2 Function Documentation

1.4.2.1 - (bool) pageEnd: (NSError ∗∗) *error*

Exits page mode.

Parameters

error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.4.2.2 - (bool) pageFillRectangle: (int) *color* error:(NSError ∗∗) *error*

Fills the current working area (or whole page if none is set) with the specified color.

Parameters

color	one of the COLOR_∗ constants
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.4.2.3 - (bool) pageFillRectangle: (int) *left* top:(int) *top* width:(int) *width* height:(int) *height* color:(int) *color* error:(NSError ∗∗) *error*

Fills a rectangle inside the current working area with specified color.

Parameters

left,top,width,height	rectangle coordinates
color	one of the COLOR_∗ constants
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.4.2.4 - (bool) pagePrint: (NSError ∗∗) *error*

Prints the content of the virtual page.

Note

The white space from the top and bottom is not printed so the print ends at the last black dot. If you want to feed the paper use the feedPaper:(int)lines function

Parameters

error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.4.2.5 - (bool) pageRectangleFrame: (int) *left* top:(int) *top* width:(int) *width* height:(int) *height* framewidth:(int) *framewidth* color:(int) *color* error:(NSError ∗∗) *error*

Draws a rectangle frame inside the current working area with specified color.

Parameters

left,top,width,height	rectangle coordinates
framewidth	width of the frame (1-64)
color	one of the COLOR_∗ constants
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.4.2.6 - (bool) pageSetWorkingArea: (int) *left* top:(int) *top* width:(int) *width* height:(int) *height* error:(NSError ∗∗) *error*

Sets a working area and orientation inside the virtual page.

No drawing can ever occur outside the said area

Parameters

left,top,width,height	working area rectangle in absolute pixels (i.e. does not depend on the page orientation)
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.4.2.7 - (bool) pageSetWorkingArea: (int) *left* top:(int) *top* width:(int) *width* heigth:(int) *height* orientation:(int) *orientation* error:(NSError ∗∗) *error*

Sets a working area and orientation inside the virtual page.

No drawing can ever occur outside the said area

Parameters

left,top,width,height	working area rectangle in absolute pixels (i.e. does not depend on the page orientation)
orientation	one of the PAGE_∗ constants
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.4.2.8 - (bool) pageStart: (NSError ∗∗) *error*

Creates a new virtual page using the maximum supported page height.

Use getInfo:(int)infocmd to get the maximum page height supported. See pageStart for more detailed information The page mode allows constructing a virtual page inside the printer, draw text, graphics, and performs some basic graphics operations (draw rectangles, frames, invert parts of the page) at any place, rotated or not, then print the result. Page mode is useful if you want to create some non-standart printout, or print vertically. Tables functions also work in page mode allowing a huge tables to be created and printed vertically.

Parameters

error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.5 Barcode Reader Functions

Functions for scanning barcodes and direct control of the barcode engine.

Functions

- (NSString ∗) - Printer::barcodeType2Text:

  Helper function to return string name of barcode type.
- (NSString ∗) - Printer::scanBarcode:timeout:error:

  Scans barcode using the built-in barcode scanning engine.

1.5.1 Detailed Description

Functions for scanning barcodes and direct control of the barcode engine.

1.5.2 Function Documentation

1.5.2.1 - (NSString ∗) barcodeType2Text: (int) *barcodeType*

Helper function to return string name of barcode type.

Parameters

barcodeType	

Returns

barcode type name

1.5.2.2 - (NSString ∗) scanBarcode: (int ∗) *barcodeType* timeout:(double) *timeout* error:(NSError ∗∗) *error*

Scans barcode using the built-in barcode scanning engine.

Parameters

barcodeType	upon success barcode type, one of the BAR_∗ constants will be stored
timeout	maximum time to wait for a barcode
error	returns error information, you can pass nil if you don't want it

Returns

barcode string or nil if function failed or no barcode was read

1.6 Magnetic Stripe Reader Functions

Functions to work with the printer's magenetic card reader.

Functions

- (NSArray ∗) - Printer::msReadCard:error:

  Reads magnetic stripe card.
- (NSDictionary ∗) - Printer::msProcessFinancialCard:track2:

  Helper function to parse financial card and extract the data - name, number, expiration date.

1.6.1 Detailed Description

Functions to work with the printer's magenetic card reader.

1.6.2 Function Documentation

1.6.2.1 - (NSDictionary ∗) msProcessFinancialCard: (NSString ∗) *track1* track2:(NSString ∗) *track2*

Helper function to parse financial card and extract the data - name, number, expiration date.

The function will extract as much information as possible.

Parameters

track1	- track1 information or nil
track2	- track2 information or nil

Returns

dictionary containing extracted data or nil if the data is invalid. Keys contained are:

"accountNumber"	Account number
"cardholderName"	Cardholder name, as stored in the card
"expirationYear"	Expiration date - year
"expirationMonth"	Expiration date - month
"serviceCode"	Service code (if any)
"discretionaryData"	Discretionary data (if any)
"firstName"	Extracted cardholder's first name
"lastName"	Extracted cardholder's last name

1.6.2.2 - (NSArray ∗) msReadCard: (double) *timeout* error:(NSError ∗∗) *error*

Reads magnetic stripe card.

For PP-60, this function is not needed - the card data is passed via delegate whenever the user swipes a card.

Parameters

timeout	timeout in seconds to read the card data. The actuall scan time may differ, but will be as close as possible to this value
error	returns error information, you can pass nil if you don't want it

Returns

String[3] containing the 3 tracks or null if timeout elapses

1.7 SmartCard Reader Functions

Functions to work with the printer's smart card reader.

Functions

- (bool) - Printer::scInit:

  Initializes and powers on the smartcard reader.
- (bool) - Printer::scClose:

  Powers down the smartcard reader.
- (NSData ∗) - Printer::scReset:

  Resets the smartcard and returns Answer To Reset.
- (NSData ∗) - Printer::scAPDU:ins:p1:p2:data:maxrcvlen:error:

  Sends an APDU command to the smartcard.

1.7.1 Detailed Description

Functions to work with the printer's smart card reader.

1.7.2 Function Documentation

1.7.2.1 - (NSData ∗) scAPDU: (int) *cla* ins:(int) *ins* p1:(int) *p1* p2:(int) *p2* data:(NSData ∗) *data* maxrcvlen:(int) *maxrcvlen* error:(NSError ∗∗) *error*

Sends an APDU command to the smartcard.

The smartcard have to be operational first by performing scInit and scReset commands on it.

Parameters

cla	The CLA parameter uint8_t
ins	The INS parameter uint8_t
p1	The P1 parameter uint8_t
p2	The P2 parameter uint8_t
data	The data buffer you want to send with the command, optional, can be null
maxrcvlen	Defines the maximum number of uint8_ts you want to receive from the smartcard. Defaults to 0
error	returns error information, you can pass nil if you don't want it

Returns

NSData with the smartcard response + 2 bytes of status or nil if command failed

1.7.2.2 - (bool) scClose: (NSError ∗∗) *error*

Powers down the smartcard reader.

Parameters

error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.7.2.3 - (bool) scInit: (NSError **) *error*

Initializes and powers on the smartcard reader.

Call this function before any other smartcard functions

Parameters

error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.7.2.4 - (NSData *) scReset: (NSError **) *error*

Resets the smartcard and returns Answer To Reset.

Call this function prior to performing APDU commands

Parameters

error	returns error information, you can pass nil if you don't want it

Returns

first byte is the protocol number (0=T0, 1=T1), the rest is ATR(Answer To Reset) data or nil if error occured

1.8 Mifare Reader Functions

Functions to work with the printer's mifare cards reader.

Functions

- (NSString ∗) - Printer::mfIdent:

  Returns mifare engine identification.
- (bool) - Printer::mfInit:

  Initializes and powers on the mifare reader module.
- (bool) - Printer::mfClose:

  Powers down mifare reader module.
- (bool) - Printer::mfRequestCards:rq1:rq2:error:

  Scans for mifare cards in the area.
- (bool) - Printer::mfAnticollision:error:

  Returns scanned card serial number.
- (bool) - Printer::mfSelectCard:sack:error:

  Select the card to use.
- (bool) - Printer::mfAuthByKey:block:key:error:

  Authenticate card block with direct key data.
- (NSData ∗) - Printer::mfRead:error:

  Reads a 16 byte block of data.
- (bool) - Printer::mfWrite:data:error:

  Writes a 16 byte block of data.
- (bool) - Printer::mfValueOperation:src_block:dst_block:value:error:

  Performs increment/decrement/restore operations.
- (bool) - Printer::mfGetReaderSerial:error:

  Returns mifare reader serial number.
- (bool) - Printer::mfWriteValue:value:error:

  Writes a 4 byte value in the card.
- (bool) - Printer::mfLoadKey:key:error:

  Stores key securely inside the mifare reader.
- (bool) - Printer::mfAuthByLoadedKey:block:keyID:error:

  Authenticate block by using previously stored key.

1.8.1 Detailed Description

Functions to work with the printer's mifare cards reader.

1.8.2 Function Documentation

1.8.2.1 - (bool) mfAnticollision: (uint32_t ∗) *serial* error:(NSError ∗∗) *error*

Returns scanned card serial number.

Parameters

serial	upon success, card serial number will be returned here
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.8.2.2 - (bool) mfAuthByKey: (char) *type* block:(int) *block* key:(uint8_t) *key[6]* error:(NSError **) *error*

Authenticate card block with direct key data.

Parameters

type	key type, either 'A' or 'B'
block	block number
key	6 bytes key
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.8.2.3 - (bool) mfAuthByLoadedKey: (char) *type* block:(int) *block* keyID:(int) *keyID* error:(NSError **) *error*

Authenticate block by using previously stored key.

Parameters

type	key type, either 'A' or 'B'
keyID	the index of the key (0-31)
block	block to authenticate
keyID	the index of the key (0-31)
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.8.2.4 - (bool) mfClose: (NSError **) *error*

Powers down mifare reader module.

Call this function after you are done with the mifare reader.

Parameters

error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.8.2.5 - (bool) mfGetReaderSerial: (uint32_t *) *serial* error:(NSError **) *error*

Returns mifare reader serial number.

Parameters

error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.8.2.6 - (NSString ∗) mfIdent: (NSError ∗∗) *error*

Returns mifare engine identification.

This is a way to query the engine and see it is there.

Parameters

error	returns error information, you can pass nil if you don't want it

Returns

identification string or nil if error occured

1.8.2.7 - (bool) mfInit: (NSError ∗∗) *error*

Initializes and powers on the mifare reader module.

Call this function before any other mifare functions.

Parameters

error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.8.2.8 - (bool) mfLoadKey: (int) *keyID* key:(uint8_t) *key[6]* error:(NSError ∗∗) *error*

Stores key securely inside the mifare reader.

The key can later be used to authenticate blocks

Parameters

keyID	the index of the key (0-31)
key	key data
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.8.2.9 - (NSData ∗) mfRead: (int) *address* error:(NSError ∗∗) *error*

Reads a 16 byte block of data.

Parameters

address	the address of the block to read
data	data buffer, where returned block will be written
error	returns error information, you can pass nil if you don't want it

Returns

received data or nil if error occured

1.8.2.10 - (bool) mfRequestCards: (bool) *allCards* rq1:(uint8_t *) *rq1* rq2:(uint8_t *) *rq2* error:(NSError **) *error*

Scans for mifare cards in the area.

Parameters

allCards	- true if you want all cards to be requested, or false for only not halted cards
rq1	(optional) upon success, the request status RQ1 will be returned here
rq2	(optional) upon success, the request status RQ2 will be returned here
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.8.2.11 - (bool) mfSelectCard: (uint32_t) *serial* sack:(uint8_t *) *sack* error:(NSError **) *error*

Select the card to use.

Parameters

serial	card serial number, received from serial
sack	(optional) SACK parameter is returned upon success
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.8.2.12 - (bool) mfValueOperation: (int) *operation* src$_b$lock : (int)src_blockdst$_b$lock : (int)dst_blockvalue : (uint32_t)valueerror : (NSError**) *error*

Performs increment/decrement/restore operations.

Parameters

operation	operation type, one if the MF_OPERATION_INCREMENT, MF_OPERATION_DECREMENT or MF_OPERATION_RESTORE
src_block	source block number
dst_block	destination block number
value	value to be incremented/decremented with
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.8.2.13 - (bool) mfWrite: (int) *address* data:(NSData ∗) *data* error:(NSError ∗∗) *error*

Writes a 16 byte block of data.

Parameters

address	the address of where to write
data	data to write in the block
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.8.2.14 - (bool) mfWriteValue: (int) *address* value:(uint32_t) *value* error:(NSError ∗∗) *error*

Writes a 4 byte value in the card.

Parameters

address	address to write to
value	the data
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.9 Table Functions

Functions to create, fill and print tables.

Functions

- (bool) - Printer::tableIsSupported

  Checks if the currently connected printer supports tables.
- (bool) - Printer::tableCreate:error:

  Create a new table using custom flags.
- (bool) - Printer::tableCreate:

  Create a new table using default settings - both horizontal and vertical borders around it.
- (bool) - Printer::tableAddColumn:

  Adds a new column using default settings - 12x24 font, plain, vertical border between the cells, left aligning.
- (bool) - Printer::tableAddColumn:error:

  Adds a new column using default settings - plain text, vertical border between the cells, left aligning.
- (bool) - Printer::tableAddColumn:style:alignment:error:

  Adds a new column using custom font and vertical border between the cells.
- (bool) - Printer::tableAddColumn:style:alignment:flags:error:

  Adds a new column.
- (bool) - Printer::tableAddCell:error:

  Adds a new cell using the font size and style and aligning of the column that cell belongs to.
- (bool) - Printer::tableAddCell:font:error:

  Adds a new cell using the font style and aligning of the column that cell belongs to.
- (bool) - Printer::tableAddCell:font:style:error:

  Adds a new cell using custom font size and style and aligning of the column that cell belongs to.
- (bool) - Printer::tableAddCell:font:style:alignment:error:

  Adds a new cell using custom font size and style and aligning.
- (bool) - Printer::tableAddDelimiter:

  Adds aa horizontal black line to the entire row that separates it from the next.
- (bool) - Printer::tableSetRowHeight:error:

  Sets the row height that will be used by default for new cells added.
- (bool) - Printer::tablePrint:

  Prints current table or throws IllegalArgumentException if cell data cannot be fit into paper.

1.9.1 Detailed Description

Functions to create, fill and print tables.

1.9.2 Function Documentation

1.9.2.1 - (bool) tableAddCell: (NSString ∗) *data* error:(NSError ∗∗) *error*

Adds a new cell using the font size and style and aligning of the column that cell belongs to.

Parameters

data	string data
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.9.2.2 - (bool) tableAddCell: (NSString ∗) *data* font:(int) *font* error:(NSError ∗∗) *error*

Adds a new cell using the font style and aligning of the column that cell belongs to.

Parameters

data	string data
font	font size, one of the FONT_size constants
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.9.2.3 - (bool) tableAddCell: (NSString ∗) *data* font:(int) *font* style:(int) *style* alignment:(int) *alignment* error:(NSError ∗∗) *error*

Adds a new cell using custom font size and style and aligning.

Parameters

data	string data
font	font size, one of the FONT_size constants
style	one or more of the font style constants (FONT_BOLD, FONT_ITALIC, etc)
alignment	date aligning, one of the ALIGN_∗ constants
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.9.2.4 - (bool) tableAddCell: (NSString ∗) *data* font:(int) *font* style:(int) *style* error:(NSError ∗∗) *error*

Adds a new cell using custom font size and style and aligning of the column that cell belongs to.

Parameters

data	string data
font	font size, one of the FONT_size constants
style	one or more of the font style constants (FONT_BOLD, FONT_ITALIC, etc)
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.9.2.5 - (bool) tableAddColumn: (NSError ∗∗) *error*

Adds a new column using default settings - 12x24 font, plain, vertical border between the cells, left aligning.

Parameters

error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.9.2.6 - (bool) tableAddColumn: (int) *font* error:(NSError ∗∗) *error*

Adds a new column using default settings - plain text, vertical border between the cells, left aligning.

Parameters

font	one of the FONT_size constants
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.9.2.7 - (bool) tableAddColumn: (int) *font* style:(int) *style* alignment:(int) *alignment* error:(NSError ∗∗) *error*

Adds a new column using custom font and vertical border between the cells.

Parameters

font	one of the FONT_size constants
style	one or more of the font style constants (FONT_BOLD, FONT_ITALIC, etc)
alignment	text alignment inside the cell, one of the ALIGN_∗ constants
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.9.2.8 - (bool) tableAddColumn: (int) *font* style:(int) *style* alignment:(int) *alignment* flags:(int) *flags* error:(NSError ∗∗) *error*

Adds a new column.

Parameters

font	one of the FONT_size constants
style	one or more of the font style constants (FONT_BOLD, FONT_ITALIC, etc)
alignment	text alignment inside the cell, one of the ALIGN_∗ constants
flags	one or more of the TABLE_BORDERS_∗ constants
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.9.2.9 - (bool) tableAddDelimiter: (NSError **) error

Adds aa horizontal black line to the entire row that separates it from the next.

Parameters

error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.9.2.10 - (bool) tableCreate: (NSError **) error

Create a new table using default settings - both horizontal and vertical borders around it.

Parameters

error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.9.2.11 - (bool) tableCreate: (int) flags error:(NSError **) error

Create a new table using custom flags.

Parameters

flags	one or more of the TABLE_BORDERS_* constants
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.9.2.12 - (bool) tableIsSupported

Checks if the currently connected printer supports tables.

Returns

TRUE if tables are supported

1.9.2.13 - (bool) tablePrint: (NSError **) error

Prints current table or throws IllegalArgumentException if cell data cannot be fit into paper.

Parameters

error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.9.2.14 - (bool) tableSetRowHeight: (int) *height* error:(NSError ∗∗) *error*

Sets the row height that will be used by default for new cells added.

Parameters

height	row height, any value less than the characters height will be auto fixed. Default is LINESPACE_DEFAULT
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.10 Cryptographic & Security Functions

Cryptographical functions - loading keys, magnetic card encryption and printer authentication.

Functions

- (NSData *) - Printer::cryptoRawGenerateRandomData:

  Generates 16 byte block of random numbers, required for some of the other crypto functions.
- (bool) - Printer::cryptoRawSetKey:encryptedData:error:
- (bool) - Printer::cryptoSetKey:key:oldKey:error:

  Used to store AES256 keys into printer's internal memory.
- (NSData *) - Printer::cryptoRawAuthenticatePrinter:error:
- (bool) - Printer::cryptoAuthenticatePrinter:error:

1.10.1 Detailed Description

Cryptographical functions - loading keys, magnetic card encryption and printer authentication. Currently that information is for the software cryptography in the printer firmware, not the hardware encrypted magnetic head.

An overview of the security, provided by the printers (see each of the crypto functions for further detail):

Firmware:

For magnetic card encryption the printer is using AES256, which is the current industry standard encryption algorithm. Magnetic card data, along with device serial number and some random bytes (to ensure every packet will be different) are being sent to the iOS program in an encrypted way.

Software:

Currently there are 2 types of keys, that can be loaded into the printer:

- AUTHENTICATION KEY - used for device authentication (for example the program can lock itself to work with very specific printer) and encryption of the firmware
- ENCRYPTION KEY - used for magnetic card data encryption. To use msr encryption, you don't need to set the AUTHENTICATION KEY.

Keys: The keys can be set/changed in two ways:

1. Using plain key data - this method is easy to use, but less secure, as it relies on program running on iPod/iPhone to have the key inside, an attacker could compromise the system and extract the key from device's memory. Call cryptoSetKey to set the keys this way. If there is an existing key of the same type inside the printer, you have to pass it too.

2. Using encrypted key data - this method is harder to implement, but provides better security - the key data, encrypted with old key data is sent from a server in secure environment to the program, running on the iOS, then the program forwards it to the printer. The program itself have no means to decrypt the data, so an attacker can't possibly extract the key. Refer to cryptoSetKey documentation for more detailed description of the loading process.

The initial loading of the keys should always be done in a secure environment.

Magnetic card encryption:

Once ENCRYPTION KEY is set, all magnetic card data gets encrypted, and is now sent via magneticCardEncryptedData instead. The printer demo program contains sample code to decrypt the data block and extract the contents - the serial number and track data.

As with keys, card data can be extracted on the iOS device itself (less secure, the application needs to have the key inside) or be sent to a secure server to be processed.

Note

The encrypted data contains printer's serial number too, this can be used for Data Origin Verification, to be sure someone is not trying to mimic data, coming from another device.

1.10.2 Function Documentation

1.10.2.1 - (bool) cryptoAuthenticatePrinter: (NSData *) *key* error:(NSError **) *error*

Note

Check out the randomData function, if you want to not use the key inside the mobile device.

Generates random data, uses the key to encrypt it, then encrypts the same data with the stored authentication key inside the printer and returns true if both data matches.

The idea: if a program wants to work with specific printer, it sets AES256 authentication key once, then on every connect the program uses key with that key. If the printer contains no key, or the key is different, the function will return FALSE.

This does not block the printer from operation, what action will be taken if devices mismatch depends on the program.

Parameters

key	32 bytes AES256 key
error	returns error information, you can pass nil if you don't want it

Returns

TRUE if the printer contains the same authentication key

1.10.2.2 - (NSData *) cryptoRawAuthenticatePrinter: (NSData *) *randomData* error:(NSError **) *error*

Note

RAW crypto functions are harder to use and require more code, but are created to allow no secret keys to reside on the device, but all the operations can be executed with data, sent from a secure server. See key if you plan to use the key in the mobile device.

Encrypts a 16 bytes block of random data with the stored authentication key and returns the result.

The idea: if a program wants to work with specific printer, it sets AES256 authentication key once, then on every connect the program generates random 16 byte block of data, encrypts it internally with the said key, then encrypts it with the printer too and compares the result. If that printer no key, or the key is different, the resulting data will totally differ from the one generated.

This does not block the printer from operation, what action will be taken if devices mismatch depends on the program.

Parameters

randomData	16 bytes block of data (presumably random bytes)
error	returns error information, you can pass nil if you don't want it

Returns

random data, encrypted with the printer's authentication key or nil if error occured

1.10.2.3 - (NSData *) cryptoRawGenerateRandomData: (NSError **) error

Generates 16 byte block of random numbers, required for some of the other crypto functions.

Parameters

error	returns error information, you can pass nil if you don't want it

Returns

16 bytes of random numbers or nil if error occured

1.10.2.4 - (bool) cryptoRawSetKey: (int) keyID encryptedData:(NSData *) encryptedData error:(NSError **) error

Note

RAW crypto functions are harder to use and require more code, but are created to allow no secret keys to reside on the device, but all the operations can be executed with data, sent from a secure server. See cryptoSetKey if you plan to use the key in the mobile device.

Used to store AES256 keys into printer's internal memory. Valid keys that can be set:

- KEY_AUTHENTICATION - if set, you can use authentication functions - randomData or key. Firmware updates will require authentication too
- KEY_ENCRYPTION - if set, magnetic card data will come encrypted via magneticCardEncryptedData

Generally the key loading process, using "Raw" commands, a program on the iOS device and a server which holds the keys will look similar to:

- (iOS program) calls cryptoRawGenerateRandomData to get 16 bytes block of random data and send these to the server
- (Server) creates byte array of 48 bytes consisting of: [RANDOM DATA: 16 bytes][KEY DATA: 32 bytes]
- (Server) if there is current encryption key set in the printer (if you want to change existing key) the server encrypts the 48 bytes block with the OLD key
- (Server) sends the result data back to the program
- (iOS program) calls cryptoRawSetKey with KEY_ENCRYPTION and the data it received from the server
- (Printer) tries to decrypt the key data if there was already key present, then extracts the key, verifies the random data and if everything is okay, sets the key

Parameters

keyID	the key type to set - KEY_AUTHENTICATION or KEY_ENCRYPTION
encryptedData	48 bytes that consists of 16 bytes random numbers received via call to cryptoRawGenerateRandomData and 32 byte AES256 key. If there has been previous key of the same type, then all 48 bytes should be encrypted with it.
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

1.10.2.5 - (bool) cryptoSetKey: (int) *keyID* key:(NSData ∗) *key* oldKey:(NSData ∗) *oldKey* error:(NSError ∗∗) *error*

Used to store AES256 keys into printer's internal memory.

Valid keys that can be set:

- KEY_AUTHENTICATION - if set, you can use authentication functions - randomData or key.
- KEY_ENCRYPTION - if set, magnetic card data will come encrypted via magneticCardEncryptedData

Parameters

keyID	the key type to set - KEY_AUTHENTICATION or KEY_ENCRYPTION
key	32 bytes AES256 key to set
oldKey	32 bytes AES256 key that was previously used, or null if there was no previous key. The old key should match the new key, i.e. if you are setting KEY_ENCRYPTION, then you should pass the old KEY_ENCRYPTION.
error	returns error information, you can pass nil if you don't want it

Returns

TRUE upon success, FALSE otherwise

Index