IPAD, DynaPro, DynaPro Go, DynaPro Mini

PIN Encryption Device
Programmer's Reference (iOS)

February 2019

Manual Part Number:
D99875654-40

REGISTERED TO ISO 9001:2015

Table 0.1 – Revisions

Rev Number	Date	Notes
1.01	Jan 31, 2014	Initial Release
2.01	Feb 12, 2014	Added MTPPSCRA Delegate Methods' return information
30	September 10, 2018	Added support for DynaPro and DynaPro Go. Updated to the latest SDK methods: loadClientCertificate, updateFirmware, getSpecialCommand, requestTipOrCashback, requestSignature, requestGetEMVTag, requestSetEMVTag, setBINTableData, requestBINTableData, getCardDataInfo, setAddress, confirmAmount, getUserDataEntry, requestPINBypass, requestClearTextUserDataEntry, sendBigBlockData, getSelectedMenuItem. Updated to the latest SDK delegates new format "deviceMessage": onNotPaired, onACKStatusComplete, onCommandStatusComplete, onDeviceInformationComplete, onClearTextUserDataEntryComplete, onSessionStart, onBINtableDataComplete, onTipOrCashBackComplete, onSelectedMenuItemComplete, onConnectionStateChanged, onCertificateError Removed methods and delegates: requestATR, onATRRequest.Complete

40	February 28, 2019	Update to correctly reference Bluetooth LE. Deprecated setDeviceType is place of setConnectionType.

SOFTWARE LICENSE AGREEMENT

IMPORTANT: YOU SHOULD CAREFULLY READ ALL THE TERMS, CONDITIONS AND RESTRICTIONS OF THIS LICENSE AGREEMENT BEFORE INSTALLING THE SOFTWARE PACKAGE. YOUR INSTALLATION OF THE SOFTWARE PACKAGE PRESUMES YOUR ACCEPTANCE OF THE TERMS, CONDITIONS, AND RESTRICTIONS CONTAINED IN THIS AGREEMENT. IF YOU DO NOT AGREE WITH THESE TERMS, CONDITIONS, AND RESTRICTIONS, PROMPTLY RETURN THE SOFTWARE PACKAGE AND ASSOCIATED DOCUMENTATION TO THE ADDRESS ON THE FRONT PAGE OF THIS DOCUMENT, ATTENTION: CUSTOMER SUPPORT.

TERMS, CONDITIONS, AND RESTRICTIONS

MagTek, Incorporated (the "Licensor") owns and has the right to distribute the described software and documentation, collectively referred to as the "Software."

LICENSE: Licensor grants you (the "Licensee") the right to use the Software in conjunction with MagTek products. LICENSEE MAY NOT COPY, MODIFY, OR TRANSFER THE SOFTWARE IN WHOLE OR IN PART EXCEPT AS EXPRESSLY PROVIDED IN THIS AGREEMENT. Licensee may not decompile, disassemble, or in any other manner attempt to reverse engineer the Software. Licensee shall not tamper with, bypass, or alter any security features of the software or attempt to do so.

TRANSFER: Licensee may not transfer the Software or license to the Software to another party without the prior written authorization of the Licensor. If Licensee transfers the Software without authorization, all rights granted under this Agreement are automatically terminated.

COPYRIGHT: The Software is copyrighted. Licensee may not copy the Software except for archival purposes or to load for execution purposes. All other copies of the Software are in violation of this Agreement.

TERM: This Agreement is in effect as long as Licensee continues the use of the Software. The Licensor also reserves the right to terminate this Agreement if Licensee fails to comply with any of the terms, conditions, or restrictions contained herein. Should Licensor terminate this Agreement due to Licensee's failure to comply, Licensee agrees to return the Software to Licensor. Receipt of returned Software by the Licensor shall mark the termination.

LIMITED WARRANTY: Licensor warrants to the Licensee that the disk(s) or other media on which the Software is recorded are free from defects in material or workmanship under normal use.

THE SOFTWARE IS PROVIDED AS IS. LICENSOR MAKES NO OTHER WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Because of the diversity of conditions and PC hardware under which the Software may be used, Licensor does not warrant that the Software will meet Licensee specifications or that the operation of the Software will be uninterrupted or free of errors.

IN NO EVENT WILL LICENSOR BE LIABLE FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE, OR INABILITY TO USE, THE SOFTWARE. Licensee's sole remedy in the event of a defect in material or workmanship is expressly limited to replacement of the Software disk(s) if applicable.

GOVERNING LAW: If any provision of this Agreement is found to be unlawful, void, or unenforceable, that provision shall be removed from consideration under this Agreement and will not affect the enforceability of any of the remaining provisions. This Agreement shall be governed by the laws of the State of California and shall inure to the benefit of MagTek, Incorporated, its successors or assigns.

ACKNOWLEDGMENT: LICENSEE ACKNOWLEDGES THAT HE HAS READ THIS AGREEMENT, UNDERSTANDS ALL OF ITS TERMS, CONDITIONS, AND RESTRICTIONS, AND AGREES TO BE BOUND BY THEM. LICENSEE ALSO AGREES THAT THIS AGREEMENT SUPERSEDES ANY AND ALL VERBAL AND WRITTEN COMMUNICATIONS BETWEEN LICENSOR AND LICENSEE OR THEIR ASSIGNS RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT.

QUESTIONS REGARDING THIS AGREEMENT SHOULD BE ADDRESSED IN WRITING TO MAGTEK, INCORPORATED, ATTENTION: CUSTOMER SUPPORT, AT THE ADDRESS LISTED IN THIS DOCUMENT, OR E-MAILED TO SUPPORT@MAGTEK.COM.

Table of Contents

SOFTWARE LICENSE AGREEMENT 4

Table of Contents 6

1 Introduction 11

1.1 About the MTPPSCRA Demo App 11

1.2 Nomenclature 11

2 How to Set Up the MagTek PIN Pad SCRA SDK 12

3 MTPPSCRA Library Methods 13

3.1 requestMSR 13

3.2 deviceReset 13

3.3 setMTPPSCRALibrary 13

3.4 cancelOperation 13

3.5 requestCommandStatus 13

3.6 stopScanningForPeripherals 13

3.7 startScanningForPeripherals 13

3.8 requestTransactionData 14

3.9 requestBypassPINCommand 14

3.10 setPan 14

3.11 endSession 14

3.12 setCommandTimeout 14

3.13 sendSpecialCommand 14

3.14 requestDeviceConfiguration 15

3.15 requestChallengeAndSessionForInformation 15

3.16 setDisplayMessage 15

3.17 sendICCAPDUWithCommand 15

3.18 sendAcquirerResponse 16

3.19 requestDeviceInformation 16

3.20 requestDeviceStatus 16

3.21 requestKeyInformation 17

3.22 requestKernelInformation 17

3.23 requestCard 17

3.24 setAmount 18

3.25 setCAPublicKey 18

3.26 requestManualCardData 18

3.27 requestSetEMVTags 19
3.28 requestUserDataEntry 19
3.29 requestResponse 20
3.30 requestPIN 21
3.31 requestSmartCard 21
3.32 openDevice 22
3.33 closeDevice 22
3.34 getDeviceType 22
3.35 isDeviceOpened 23
3.36 isDeviceConnected 23
3.37 getConnectedPeripheral 23
3.38 getDiscoveredPeripherals 23
3.39 setDeviceUUIDString 23
3.40 setDeviceProtocolString 23
3.41 clearBuffer 24
3.42 getStatusCode 24
3.43 getPAN 24
3.44 getKSN 24
3.45 getSDKVersion 24
3.46 getEncodeType 24
3.47 getProductID 25
3.48 getDeviceModel 25
3.49 getDeviceSerial 25
3.50 getDeviceFirmwareVersion 25
3.51 getTrack1 25
3.52 getTrack2 25
3.53 getTrack3 26
3.54 getTrack1Masked 26
3.55 getTrack2Masked 26
3.56 getTrack3Masked 26
3.57 getMaskedTracks 26
3.58 getTrack2Equivalent 26
3.59 getMagnePrint 27
3.60 getMadnePrintStatus 27
3.61 getTrack1DecodeStatus 27

3.62 getTrack2DecodeStatus 27
3.63 getTrack3DecodeStatus 27
3.64 getLastName 27
3.65 getFirstName 27
3.66 getMiddleName 28
3.67 getExpDate 28
3.68 getCardType 28
3.69 getCardLast4 28
3.70 getCardServiceCode 28
3.71 getResponseData 28
3.72 getSREDResponseData 29
3.73 getTransactionStatus 29
3.74 isSignatureRequired 29
3.75 isDeviceSRED 29
3.76 getEPB 29
3.77 getPINKSN 29
3.78 getPINStatusCode 30
3.79 getSelectedMenuItem 30
3.80 setConnectionType 31
3.81 loadClientCertificate 31
3.82 updateFirmware 31
3.83 getSpecialCommand 31
3.84 requestTipOrCashback 32
3.85 requestSignature 33
3.86 requestGetEMVTag 33
3.87 setBINTableData 34
3.88 requestBINTableData 34
3.89 getCardDataInfo 34
3.90 setAddress 34
3.91 confirmAmount 34
3.92 getUserDataEntry 35
3.93 requestPINBypass 35
3.94 requestClearTextUserDataEntry 35
3.95 sendBigBlockData 36
4 MTPPSCRA Delegate Methods 36

4.1 deviceDataAvailable 36
4.2 deviceMessageChange 36
4.3 deviceMessageDataChanged 36
4.4 onDataArriveCompleteEvent 36
4.5 onSendICCAPDUComplete 36
4.6 deviceMessageCAPublicKey 37
4.7 deviceMessagePinData 37
4.8 deviceMessageUserSelection 37
4.9 deviceMessageDisplayMessageDone 37
4.10 onBypassPINCommandComplete 37
4.11 onSendCommandTimeOutComplete 37
4.12 deviceMessageDeviceConfiguration 38
4.13 deviceMessageARQCRequest 38
4.14 deviceMessageEMVData 38
4.15 onRequestPowerUpResetICCAPDUComplete 38
4.16 onRequestChallengeAndSessionKeyRequestComplete 38
4.17 onCardRequestComplete 38
4.18 deviceMessageCardStatus 39
4.19 deviceMessageEMVTags 39
4.20 onRequestSmartCardComplete 39
4.21 onSendAcquirerResponseComplete 39
4.22 deviceMessageUserDataEntry 39
4.23 deviceMessageKernelInformation 40
4.24 deviceNotPaired 40
4.25 deviceMessageACKStatus 40
4.26 deviceMessageClearTextUserEntry 40
4.27 deviceMesageDeviceInformation 40
4.28 deviceMessageUserSignature 40
4.29 deviceMessageDeviceStatus 40
4.30 deviceMessageKeyInformation 41
4.31 deviceMessageBINTableData 41
4.32 deviceMessageTipOrCashback 41
4.33 deviceMessageSelectedMenuItem 41
4.34 onConnectionStateChanged 42
4.35 onCertificateError 42

1 - Table of Contents

Appendix A Code Examples 43
A.1 Instantiating the MTPPSCRA Library 43
A.2 Open Device 43
A.3 Close Device 43

1 Introduction

This document provides instructions for software developers who want to create software solutions that include a DynaPro connected via Ethernet, DynaPro Go connected via 802.11 Wireless or Bluetooth LE, DynaPro Mini connected via Bluetooth LE or Apple 30-pin connector. It is part of a larger library of documents designed to assist DynaPro Mini implementers, which includes:

- ***D99875586 DynaPro Installation and Operation Manual***
- ***D99875642 DynaPro Mini Installation and Operation Manual***
- ***D99875622 Dynapro Image Installation Guide***
- ***D99875585 DynaPro Programmer's Reference (Commands)***
- ***D99875629 DynaPro Mini Programmer's Reference (Commands)***
- ***D998200136 DynaPro Go Programmer's Manual (Commands)***
- ***D99875656 IPAD, DynaPro, DynaPro Go, DynaPro Mini Programmer's Reference (C++)***
- ***D99875668 IPAD, DynaPro, DynaPro Go, DynaPro Mini Programmer's Reference (Android)***
- ***D99875633 IPAD, DynaPro, DynaPro Go, DynaPro Mini Programmer's Reference (Java / Java Applet)***
- ***D99875654 IPAD, DynaPro, DynaPro Go, DynaPro Mini Programmer's Reference (iOS)***

1.1 About the MTPPSCRA Demo App

The MTPPSCRA Demo app, available from MagTek, provides demonstration source code and reusable MTPPSCRA SDK libraries that provide custom software solutions an easy-to-use interface with DynaPro Mini. Developers of custom-branded software that include the MTPPSCRA SDK libraries can then make their software available to customers on the Apple App Store, or distribute it internally as part of an enterprise solution.

1.2 Nomenclature

The general terms “device” and “host” are used in different, often incompatible ways in a multitude of specifications and contexts. For example “host” may have different meanings in the context of USB communication than it does in the context of networked financial transaction processing. In this document, “device” and “host” are used strictly as follows:

- **Device** refers to the Pin Entry Device (PED or PIN pad) that receives and responds to the command set specified in this document; in this case, DynaPro Mini.
- **Host** refers to the piece of general-purpose electronic equipment the device is connected or paired to, which can send data to and receive data from the device. Host types include PC and Mac computers/laptops, tablets, smartphones, teletype terminals, and even test harnesses. In many cases the host may have custom software installed on it that communicates with the PED. When “host” must be used differently, it is qualified as something specific, such as “USB host.”

Because the Bluetooth LE communication layer uses a very specific meaning for the term “Application,” this document favors the term **software** for software on the host that provides an interface for the operator, such as a cashier. The combination of device(s), host(s), software, firmware, configuration settings, physical mounting and environment, user experience, and documentation is referred to as the **solution**.

Similarly, the word “user” is used in different ways in different contexts. In this document, **user** generally refers to the **cardholder**.

2 How to Set Up the MagTek PIN Pad SCRA SDK

To add the MTPPSCRA SDK libraries to a custom software project in the XCode development environment, follow these steps:

1) Download the MTPPSCRA Demo app from MagTek.com.
2) Open your custom software project in XCode.
3) Open the MTPPSCRA Demo app folder in Finder.
4) Open the `Library` subfolder.
5) Include the following files in your custom software project within XCode:
   a) `libMTPPSCRA.a`
   b) `MTPPSCRA.h`
   c) `MTObject.h`
6) Ensure the library search paths are set up correctly.
7) Clean, build, and run your custom software project to make sure the library imported correctly.
8) In your custom software, create an instance of `MTPPSCRA`. For examples, see the source code included with the MTPPSCRA Demo app and / or **Appendix A Code Examples**.
9) Begin using the features provided by the MTPPSCRA object's methods. For details about these methods, see section 3 **MTPPSCRA Library Methods**.

3 MTPPSCRA Library Methods

After creating an instance of the MTTPSCRA object in your custom software project (see section 2 **How to Set Up the MagTek PIN Pad SCRA SDK**), use the methods described in this section to communicate with DynaPro Mini.

3.1 requestMSR

This method retrieves the available MSR Data.

```
- (void)requestMSR
```

3.2 deviceReset

This method simply sends the reset command to the device.

```
- (int)deviceReset
```

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.3 setMTPPSCRALibrary

This method initializes the MTPPSCRA Library.

```
- (void)setMTPPSCRALibrary
```

3.4 cancelOperation

This method sends the cancel command to the device.

```
- (int)cancelOperation
```

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.5 requestCommandStatus

This method creates a separate synchronous thread that loops until the device returns response status data. If no response arrives within the timeout specified by **setCommandTimeout**, the method will time out.

```
- (NSInteger)requestCommandStatus
```

Return Value:
Returns an `NSInteger` value (0: Success, Non-Zero: Error)

3.6 stopScanningForPeripherals

This method stops the scanning for surrounding Peripherals.

```
- (void)stopScanningForPeripherals
```

3.7 startScanningForPeripherals

This method starts the scanning for surrounding Peripherals.

```
- (void)startScanningForPeripherals:(void(^)(NSArray*))handler
```

3.8 requestTransactionData

This method sends the `01AB` command to the device for the Transaction Data.

```
- (NSInteger)requestTransactionData
```

3.9 requestBypassPINCommand

This sends the Bypass PIN command string to the device.

```
- (NSInteger)requestBypassPINCommand
```

3.10 setPan

This method sends PAN data to the device.

```
- (int)setPAN:(NSString *)pan
```

Parameter	Description
pan	value for PAN data (8 - 19 ASCII digits).

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.11 endSession

This method clears all existing session data.

```
- (int)endSession:(Byte)messageID
```

Parameter	Description
messageID	value between 0 - 4 indicating the message to display

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.12 setCommandTimeout

This method sets the command timeout interval.

```
- (NSInteger)setCommandTimeOut:(NSTimeInterval)interval
```

Parameter	Description
interval	value for setting the interval

Return Value:
Returns an `NSInteger` value (0: Success, Non-Zero: Error)

3.13 sendSpecialCommand

This method sends a custom “SET” command to the device.

```
- (int)sendSpecialCommand(NSData *)commandIn
```

Parameter	Description
commandIn	Value for the command to send.

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.14 requestDeviceConfiguration

This method sends the Request Device Information report to the device.

```
- (int)requestDeviceConfiguration
```

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.15 requestChallengeAndSessionForInformation

This method returns an `NSString` with the requested feature report information.

```
- (NSInteger)requestChallengeAndSessionKeyForInformation:(NSString
**)information
```

Parameter	Description
information	reference value for the information requested

Return Value:
Returns an `NSInteger` value (0: Success, Non-Zero: Error)

3.16 setDisplayMessage

This method will show a predefined message on the device's LCD display.

```
- (NSInteger)setDisplayMessage:(Byte)waitTime
             messageID:(Byte)messageID
```

Parameter	Description
waitTime	value for how long the message will be displayed.
messageID	value for the predefined message to be displayed.

Return Value:
Returns an `NSInteger` value (0: Success, Non-Zero: Error)

3.17 sendICCAPDUWithCommand

This method sends the ICC APDU report to the device.

```
- (NSInteger)sendICCAPDUWithCommand:(unsigned char *)command
             withCommandLength:(NSInteger)commandLength
```

Parameter	Description
command	value for the command to send
commandLength	value for the command value length

Return Value:
Returns an `NSInteger` value (0: Success, Non-Zero: Error)

3.18 sendAcquirerResponse

This method sends the Acquirer report to the device.

```
- (int)sendAcquirerResponse:(NSData *)responseData
```

Parameter	Description
responseData	value containing the Acquirer Response data

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.19 requestDeviceInformation

This method will retrieve the requested device information.

```
- (NSInteger)requestDeviceInformation:(unsigned char)mode
            information:(NSString *)information
```

Parameter	Description
mode	Information requested: 0 – Product_ID 1 – Maximum Application Message Size 2 – Capability String 3 – Manufacturer 4 – Product Name 5 – Serial Number 6 – Firmware Number 7 – Build Info 8 – MAC address for ethernet versions only
information	value containing the requested information returned from the device.

Return Value:
Returns an `NSInteger` value (0: Success, Non-Zero: Error)

3.20 requestDeviceStatus

This method sends a commond to retrieve the device status information.

```
- (int)requestDeviceStatus
```

Return Value:

Returns an `int` value (0: Success, Non-Zero: Error)

3.21 requestKeyInformation

This method will retrieve the device's key information.

```
- (int)requestKeyInformation:(Byte)information
```

Parameter	Description
information	Value containing the key information ID.

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.22 requestKernelInformation

This method will retrieve the device's kernel information.

```
- (NSInteger)requestKernalInformation:(Byte)kernelInfoID
```

Parameter	Description
kernelInfoID	value containing the kernel information ID: 0x00 – Version L1 Kernel 0x01 – Version L2 Kernel 0x02 – Checksum/Signature L1 Kernel 0x03 – Checksum/Signature L2 Kernel 0x03 – Checksum/Signature L2 Kernel + Configuration

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.23 requestCard

This method triggers the device to begin the Card Swipe Transaction.

```
- (int)requestCard:(Byte)waitTime
            messageID:(Byte)messageID
            beepTones:(Byte)beepTones
```

Parameter	Description
waitTime	value for the amount of time the user has to complete a card swipe
messageID	value for the message to prompt the user with: 0x00 - CardMsgSwipeCardIdle 0x01 - CardMsgSwipeCard 0x02 - CardMsgPleaseSwipeCard 0x03 - CardMsgPleaseSwipeAgain

Parameter	Description
beepTones	Value for the tone to be used: 0x00 – No Sound 0x01 – Single Beep 0x02 – Double Beeps

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.24 setAmount

This method sets the amount before beginning a transaction process.

```
- (int)setAmount:(NSString *)amount
```

Parameter	Description
amount	value for the amount to be used for the transaction

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.25 setCAPublicKey

This method sets/deletes the corresponding CA Public key given the type of operation.

```
- (int)setCAPublicKey:(Byte)operation
             withCAPublicKeyBlock:(NSData*)CAPublicKeyBlock
```

Parameter	Description
operation	value for the type of operation to be performed: 0 – Erase all CA Public Keys 1 – Erase all CA Public Keys for a given RID 2 – Erase a single CA Public Key 3 – Add a single CA Public Key 0x0F – Read all CA Public Keys
CAPublicKeyBlock	value for the CA Public Key to be sent

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.26 requestManualCardData

This method triggers the device to begin a manual card data entry transaction.

```
- (int)requestManualCardData:(Byte)waitTime
             beepTones:(Byte)beepTones
             options:(Byte)options
```

Parameter	Description
waitTime	value for the amount of time the user has to begin the Manual Data entry
beepTones	Value for the tone to be used: 0x00 – No Sound 0x01 – Single Beep 0x02 – Double Beeps
options	value for the option to be used: Default is used here

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.27 requestSetEMVTags

This method sends the EMV Tag report to the device to get or set EMV Tags.

```
- (int)requestSetEMVTags:(Byte)tagType
          tagOperation:(Byte)tagOperation
          inputTLVData:(NSData *)inputTLVData
          database:(Byte)database
          option:(Byte)option
          reserved:(Byte*)reserved
```

Parameter	Description
tagType	value for the EMV Tag to set or get: 0x00 – Reader Tags
tagOperation	value for the type of operation to be performed: 1 – Write Operation 0xFF – Set to Factory defaults
inputTLVData	value for TLV Data Block to be sent to the device.
option	value for response type. 0 – Normal Response 1 – Delay Response
reserved	value for the reserved bytes.

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.28 requestUserDataEntry

This method sends the User Data Entry report to the device. The device will prompt the user to enter SSN, zip code, or birth date by displaying one of four preset messages.

```
- (int)requestUserDataEntry:(Byte)waitTime
            messageID:(Byte)messageID
            beepTone:(Byte)beepTones
```

Parameter	Description
waitTime	value for the amount of time the user has to begin the Data entry
messageID	value for the message to prompt the user with: 0 – SSN, 1 – Zip Code, 2 – Birth (Four Year), 3 – Birth (Two Year)
beepTones	Value for the tone to be used: 0x00 – No Sound 0x01 – Single Beep 0x02 – Double Beeps

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.29 requestResponse

This method sends a command to prompt the user to select a transaction type.

```
- (NSInteger)requestResponse:(Byte)waitTime
             selectMsg:(Byte)selectMsg
             keyMask:(Byte)keyMask
             beepTones:(Byte)beepTones
```

Parameter	Description
waitTime	value for the amount of time the user has to select the option
selectMsg	value for the message to prompt the user with: 0 – Transaction Type (Credit/Debit), 1 – Verify Transaction Amount, 2 – Credit Other Debit, 3 – Credit EBT Debit, 4 – Credit Gift Debit, 5 – EBT Gift Other
keyMask	value for key codes to mask: 8 – Enter, 4 – Right, 2 – Middle, 1 – Left. These values can be combined using OR
beepTones	Value for the tone to be used: 0x00 – No Sound 0x01 – Single Beep 0x02 – Double Beeps

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.30 requestPIN

This method sends the Request User PIN report to the device. The device will prompt the user to enter a PIN.

```
- (int)requestPIN:(Byte)waitTime
            pinMode:(Byte)pinMode
            maxPINLength:(Byte)maxPINLength
            minPINLength:(Byte)minPINLength
            beepTones:(Byte)beepTones
            option:(Byte)option
```

Parameter	Description
waitTime	value for the amount of time the user has req select the option
pinMode	value for the message to prompt the user with: 0 – PinMsgEnterPin 1 – PinMsgEnterPinAmt 2 – PinMsgReenterPINAmt 3 – PinMsgReenterPIN 4 – PinMsgVerifyPIN
maxPINLength	value for maximum PIN length that can be entered: Must be less than 13
minPInLength	value for minimum PIN length that can be entered: Must be less greater than 3
beepTones	Value for the tone to be used: 0x00 – No Sound 0x01 – Single Beep 0x02 – Double Beeps
option	value for the option to verify or not to verify the PIN: 0 – ISO0 Format No verify PIN 1 – ISO3 Format No verify PIN 2 – ISO0 Format Verify PIN 3 – ISO3 Format

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.31 requestSmartCard

This method triggers a Smart Card transaction.

```
- (int)requestSmartCard:(Byte)cardType
            confirmationWaitTime:(Byte)confirmationWaitTime
            pinEnteringTime:(Byte)pinEnteringTime
            beepTones:(Byte)beepTones
            option:(Byte)option
            amount:(Byte *)amount
            transactionType:(Byte)transactionType
            cashBack:(Byte *)cashBack
            reserved:(NSData *)reserved
```

Parameter	Description
cardType	value for the card type that can be used for the transaction: 1 – Magnetic Stripe 2 – Contact Smart Card 3 – Magnetic Stripe or Contact Smart Card
confirmationWaitTime	value for the amount of time the user has to begin the transaction
pinEnteringTime	value for the amount of time the user has to enter the PIN
beepTones	Value for the tone to be used: 0x00 – No Sound 0x01 – Single Beep 0x02 – Double Beeps
option	value for the option to be used: 0 – Normal 1 – Bypass PIN 2 – Force Online 4 – Acquirer not available
amount	value for the amount to be used and authorized
transactionType	value for the type of transaction to be used
cashback	value for the amount of cash back to be used
reserved	value for the reserved bytes

Return Value:
Returns an `int`value (0: Success, Non-Zero: Error)

3.32 openDevice

This method opens the connection to the DynaPro Mini.

```
- (int)openDevice
```

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.33 closeDevice

This method closes the connection to the DynaPro Mini.

```
- (int)closeDevice
```

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.34 getDeviceType

This method retrieves the device type.

```
- (int)getDeviceType
```

Return Type:
Returns the device type.

3.35 isDeviceOpened

This method retrieves the device’s open status.

```
- (BOOL)isDeviceOpened
```

Return Value:
YES if the device communication is opened, otherwise NO.

3.36 isDeviceConnected

This method retrieves the device’s connection status.

```
- (BOOL)isDeviceConnected
```

Return Value:
YES if the device is connected, otherwise NO.

3.37 getConnectedPeripheral

This method retrieves the currently connected peripheral (DynaPro Mini)

```
- (CBPeripheral *)getConnectedPeripheral
```

Return Value:
Returns the currently connected peripheral’s information.

3.38 getDiscoveredPeripherals

This method retrieves the surrounding discovered peripherals(DynaPro Minis)

```
- (NSMutableArray *)getDiscoveredPeripherals
```

Return Value:
Returns a mutable array of the discovered peripherals.

3.39 setDeviceUUIDString

This method sets the UUID string for the device to be connected to through Bluetooth LE.

```
- (void)setDeviceUUIDString:(NSString *)uuidString
```

Parameter	Description
uuidString	Value for the UUID string of the device to be connected to. This is only for devices supporting Bluetooth LE interface mode only.

3.40 setDeviceProtocolString

This method sets the protocol string for the connected device’s session to be opened through UART.

```
- (void)setDeviceProtocolString:(NSString *)protocolString
```

Parameter	Description
protocolString	Value for the protocol string to open the device's session. This is for the DynaPro Mini UART (30-PIN) only.

3.41 clearBuffer

This method clears all of the buffers that are stored during interactions with the device.

```
- (void)clearBuffer
```

3.42 getStatusCode

This method retrieves the current status of the issued report.

```
- (int)getStatusCode
```

Return Value:
Returns the current status of the device after issuing a command.

3.43 getPAN

This method retrieves the PAN

```
- (NSString *)getPAN
```

Return Value:
Returns the stored PAN string.

3.44 getKSN

This method retrieves the KSN

```
- (NSString *)getKSN
```

Return Value:
Returns the stored KSN string.

3.45 getSDKVersion

This method retrieves the current version of the SDK.

```
- (NSString *)getSDKVersion
```

Return Value:
Returns the current SDK version string.

3.46 getEncodeType

This method retrieves the encode type of a card after a card swipe.

```
- (NSString *)getEncodeType
```

Return Value:
Returns the encode type string.

3.47 getProductID

This method retrieves the product ID of the device.

```
- (NSString *)getProductID
```

Return Value:
Returns the product ID of the device.

3.48 getDeviceModel

This method retrieves the model number of the device.

```
- (NSString *)getDeviceModel
```

Return Value:
Returns the model number string.

3.49 getDeviceSerial

This method retrieves the serial number of the device.

```
- (NSString *)getDeviceSerial
```

Return Value:
Returns the serial number string.

3.50 getDeviceFirmwareVersion

This method retrieves the firmware version of the device.

```
- (NSString *)getDeviceFirmwareVersion
```

Return Value:
Returns the firmware version string.

3.51 getTrack1

This method retrieves the encrypted track one data.

```
- (NSString *)getTrack1
```

Return Value:
Returns the encrypted track one data string.

3.52 getTrack2

This method retrieves the encrypted track two data.

```
- (NSString *)getTrack2
```

Return Value:
Returns the encrypted track two data string.

3.53 getTrack3

This method retrieves the encrypted track three data.

```
- (NSString *)getTrack3
```

Return Value:
Returns the encrypted track three data string.

3.54 getTrack1Masked

This method retrieves the masked track one data.

```
- (NSString *)getTrack1Masked
```

Return Value:
Returns the masked track one data string.

3.55 getTrack2Masked

This method retrieves the masked track two data.

```
- (NSString *)getTrack2Masked
```

Return Value:
Returns the masked track two data string.

3.56 getTrack3Masked

This method retrieves the masked track three data.

```
- (NSString *)getTrack3Masked
```

Return Value:
Returns the masked track three data string.

3.57 getMaskedTracks

This method retrieves all of the masked tracks' data.

```
- (NSString *)getMaskedTracks
```

Return Value:
Returns the masked tracks' data string.

3.58 getTrack2Equivalent

This method retrieves the track two equivalent data from a smart card. This is only possible if the device's EMV Tags have been properly configured to extract the track two equivalent data TLV tag.

```
- (NSString *)getTrack2Equivalent
```

Return Value:
Returns the track two equivalent data string.

3.59 getMagnePrint

This method retrieves the MagnePrint data.

```
- (NSString *)getMagnePrint
```

Return Value:
Returns the MagnePrint data string.

3.60 getMadnePrintStatus

This method retrieves the MagnePrint status.

```
- (NSString *)getMagnePrintStatus
```

Return Value:
Returns the MagnePrint status string.

3.61 getTrack1DecodeStatus

This method retrieves the track one decode status.

```
- (NSString *)getTrack1DecodeStatus
```

Return Value:
Returns the track one decode status string.

3.62 getTrack2DecodeStatus

This method retrieves the track two decode status.

```
- (NSString *)getTrack2DecodeStatus
```

Return Value:
Returns the track two decode status string.

3.63 getTrack3DecodeStatus

This method retrieves the track three decode status.

```
- (NSString *)getTrack3DecodeStatus
```

Return Value:
Returns the track three decode status string.

3.64 getLastName

This method retrieves the last name data.

```
- (NSString *)getLastName
```

Return Value:
Returns the last name data string.

3.65 getFirstName

This method retrieves the first name data.

```
- (NSString *)getFirstName
```

Return Value:
Returns the first name data string.

3.66 getMiddleName

This method retrieves the middle name data.

```
- (NSString *)getMiddleName
```

Return Value:
Returns the middle name data string.

3.67 getExpDate

This method retrieves the expiration date data.

```
- (NSString *)getExpDate
```

Return Value:
Returns the expiration date data string.

3.68 getCardType

This method retrieves the card type data.

```
- (NSString *)getCardType
```

Return Value:
Returns the card type data string.

3.69 getCardLast4

This method retrieves the last four digits of the card number.

```
- (NSString *)getCardLast4
```

Return Value:
Returns the card number’s last four digits as a string.

3.70 getCardServiceCode

This method retrieves the card service code data.

```
- (NSString *)getCardServiceCode
```

Return Value:
Returns the card service code data string.

3.71 getResponseData

This method retrieves the response data that was received from the device.

```
- (NSString *)getResponseData
```

Return Value:
Returns the response data string.

3.72 getSREDResponseData

This method retrieves the response data received after a smart card transaction on a device with SRED firmware. This information can then be decrypted as needed.

```
- (NSString *)getSREDResponseData
```

Return Value:
Returns the SRED response data string.

3.73 getTransactionStatus

This method retrieves the transaction status of the current transaction.

```
- (NSString *)getTransactionStatus
```

Return Value:
Returns the transaction status string.

3.74 isSignatureRequired

This method will determine if a smart card is a chip and PIN or chip and signature card.

```
- (BOOL)isSignatureRequired
```

Return Value:
YES if the current smart card being used is a chip and signature card. NO if the current smart card being used is a chip and PIN(the device will handle the chip and PIN card).

3.75 isDeviceSRED

This method will determine if the device's firmware is SRED or Non-SRED.

```
- (BOOL)isDeviceSRED
```

Return Value:
YES if the firmware is SRED. NO if the firmware is Non-SRED.

3.76 getEPB

This method retrieves the encrypted PIN blick data after issuing **requestPIN**.

```
- (NSString *)getEPB
```

Return Value:
Returns the stored encrypted PIN block data string.

3.77 getPINKSN

This method retrieves the PIN KSN

```
- (NSString *)getPINKSN
```

Return Value:
Returns the stored PIN KSN string.

3.78 getPINStatusCode

This method retrieves the PIN status code.

```
- (NSString *)getPINStatusCode
```

Return Value:
Returns the stored PIN status code string.

3.79 getSelectedMenuItem

This method directs the device to present a list of menu items for user to select from. The application should first call sendBigBlockData method with menu items before calling this method.

```
- (int)getSelectedMenuItem:(Byte)waitTime
       mode:(Byte)mode
       tone:(Byte)tone;
```

Parameter	Description
waitTime	Wait time in seconds to make a menu selection.
mode	Values: 0x00 – Selection Table 0x01 – Selection Bill
tone	Value for the tone to be used: 0x00 – No Sound 0x01 – Single Beep 0x02 – Double Beeps

Returns an `NSInteger` value (0: Success, Non-Zero: Error).

3.80 setConnectionType

This method sets the type of connection to the device.

```
- (void)setConnectionType: (ConnectionType)connection
```

Parameter	Description

connection	Value for the connection type to open:
	Unknown = 0,
	Audio = 1,
	BluetoothLE = 2,
	BLEEMV = 3,
	Bluetooth = 4,
	HID = 5,
	Serial = 6,
	Net = 7,
	Net_TLS12 = 8,
	Net_TLS12_Trust_All = 9

3.81 loadClientCertificate

This method loads a client certificate for mutual authentication over wireless connection.

```
- (int)loadClientCertificate:(NSString*)format
            data:(NSData*)data
            password:(NSString*)string;
```

Parameter	Description
format	Format of the certificate file. Use: "PKCS12"PKCS12
data	Data for the certificate or certificate chain. Data format is PKCS12.
password	Password for the data.

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.82 updateFirmware

This method updates the device's firmeware.

```
- (int)updateFirmware:(NSData*)firmwareData;
```

Parameter	Description
firmwareData	Binary data of firmware.

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.83 getSpecialCommand

This method sends a custom "GET" command to the device.

```
- (int)getSpecialCommand:(NSData*)commandIn;
```

Parameter	Description
commandIn	Value for the command to send.

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.84 requestTipOrCashback

This method directs the device to request tip or cashback information from user.

```
- (int)requestTipOrCashback:(Byte)waitTime
      mode:(Byte)mode
      tone:(Byte)tone
      amount:(Byte*)amount
      taxAmount:(Byte*)taxAmount
      taxRate:(Byte*)taxRate
      tipMode:(Byte)tipMode
      option1:(Byte)option1
      option2:(Byte)option2
      option3:(Byte)option3
      reserved:(Byte*)reserved;
```

Parameter	Description
waitTime	Time in seconds to complete the operation.
mode	Operating mode: 0x00 – Tip Mode 0x01 – Cashback Mode
tone	Value for the tone to be used: 0x00 – No Sound 0x01 – Single Beep 0x02 – Double Beeps
amount	Value for the amount to be used for the transaction (6 bytes).
taxAmount	Value for the amount to be used for the calculated tax (6 bytes).
taxRate	Value for the amount to be used for the tax rate percentage (3 bytes).
tipMode	Fixed Tip Selection mode: 0x00 – Percent Mode 0x01 – Amount Mode
option1	Value of the percent or amount shown on the left side of the display above the selection buttons.
option2	Value of the percent or amount shown in the middle of the display above the selection buttons.

Parameter	Description
option3	Value of the percent or amount shown on the right side of the display above the selection buttons.
reserved	Value of the reserved data.

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.85 requestSignature

This method sends a command to request the user's signature.

```
- (int)requestSignature:(Byte)waitTime
      beepTones:(Byte)beepTones
      options:(Byte)options;
```

Parameter	Description
waitTime	Time in seconds the user has to complete signature.
beepTones	Value for the tone to be used: 0x00 – No Sound 0x01 – Single Beep 0x02 – Double Beeps
options	Options to be used: 0x00 – Timeout clears any signature data 0x01 – Timeout returns timeout status plus length collected. Sig Data can be requested.

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.86 requestGetEMVTag

This method retrieves EMV Tag data.

```
- (int)requestGetEMVTag:(Byte)tagType
      tagOperation:(Byte)tagOperation
      inputTLVData:(NSData*)inputTLVData;
```

Parameter	Description
tagType	Value for the EMV Tag to get: 0x00 – Reader Tags 0x10 – Terminal Tags 0x11 – Contactless DRL Tags (payWave only)
tagOperation	Value for the type of operation to be performed: 0x00 – Read specific tags 0x0F – Real all tags

intputTLVData	Value for TLV Data tags list to be retrieved from the device.

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.87 setBINTableData

This method sends a command to the device to set the BIN Table data.

```
- (int)setBINTableData:(NSData*)binTable
```

Parameter	Description
binTable	Value of the BIN table data to be sent to the device.

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.88 requestBINTableData

This method sends a request to retrieve BIN table data from the device.

```
- (int)requestBINTableData
```

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.89 getCardDataInfo

This method returns the card data information.

```
- (CardDataInfo*)getCardDataInfo
```

Return Value:
Returns an `CardDataInfo` containing the detailed information of the card.

3.90 setAddress

This method sets the address for the connection to the device.

```
- (void)setAddress:(NSString*)address
```

Parameter	Description
address	String value of the address.

3.91 confirmAmount

This method sends command to prompt the user to confirm the amount for the transaction.

```
- (int)confirmAmount:(Byte)waitTime beepTones:(Byte)beepTones
```

Parameter	Description
waitTime	Time in seconds the user has to select the option.

Parameter	Description
beepTones	Value for the tone to be used 0x00 – No Sound 0x01 – Single Beep 0x02 – Double Beeps

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error)

3.92 getUserDataEntry

This method returns the data of the user data entry.

```
- (UserDataEntry*)getUserDataEntry:(BOOL)clearBuffer
```

Parameter	Description
clearBuffer	Boolean value indicating whether to clear the data of the user data entry.

Return Value:
Returns `UserDataEntry` containing the data of the user data entry.

3.93 requestPINBypass

This method sends the Bypass PIN command to the device. This affects the behavior of **requestSmartCard**.

```
- (int)requestPINBypass
```

Return Value:
Returns an `int` value (0: Success, Non-Zero: Error).

3.94 requestClearTextUserDataEntry

This method sends a command to prompt the user to enter SSN, Zip code, Birthdate, Activation code, Server/Waiter code, or Ticket Number.

```
- (int)requestClearTextUserDataEntry:(Byte)waitTime
      messageID:(Byte)messageID
      beepTones:(Byte)beepTones;
```

Parameter	Description
waitTime	Time in seconds the user has to begin the data entry.
messageID	Cardholder data mode: 0x00 – Enter SSN (9 digits) 0x01 – Enter Zip code (5 digits) 0x02 – Enter Birthdate (8 digits, in MM/DD/YYYY format) 0x03 – Enter Birthdate (6 digits, in MM/DD/YY format) 0x04 – Enter Activation code (4 digits) (Activation Codes only) 0x05 – Enter Server/Waiter code (4 to 9 digits) (Handheld Operation Only) 0x06 – Enter Ticket Number (4 to 9 digits) (Handheld Operation Only)

Parameter	Description
beepTones	Value for the tone to be used: 0x00 – No Sound 0x01 – Single Beep 0x02 – Double Beeps

Returns an `int` value (0: Success, Non-Zero: Error).

3.95 sendBigBlockData

This method sends a packet of big block data to the device.

```
- (int)sendBigBlockData:(Byte)command
      dataBlock:(NSData*)dataBlock
      completion:(void (^)(void))completion
```

Parameter	Description
command	The command ID related to the big block data to be sent.
dataBlock	Byte array value of the data to be sent.
completion	After send big block is done, completion callback will come here.

Returns an `int` value (0: Success, Non-Zero: Error).

4 MTPPSCRA Delegate Methods

After issuing the methods in section 3 **MTPPSCRA Library Methods,** the MTPPSCRA SDK libraries will call these Delegate methods (callback functions) to provide the requested data and / or a detailed response.

4.1 deviceDataAvailable

This delegate will be called when data is ready to be retrieved after issuing a command.

4.2 deviceMessageChange

This delegate will be called when there is a change in device state.

4.3 deviceMessageDataChanged

This delegate will be called when data arrives from device.

4.4 onDataArriveCompleteEvent

This delegate will be called when data is returned from **sendSpecialCommand**.

Parameter	Description
data	Byte array value is returned containing the data received from the device

4.5 onSendICCAPDUComplete

This delegate will be called when data is returned from sendICCAPDUWithCommand.

Parameter	Description
data	A String value is returned containing the ICC APDU data received from the device

4.6 deviceMessageCAPublicKey

This delegate will be called when data is returned from setCAPublicKey**.**

Parameter	Description
keyData	Byte array value is returned containing the CA Public Key data received from the device.

4.7 deviceMessagePinData

This delegate will be called when data is returned from **requestPIN**.

Parameter	Description
operationStatus	Status of the operation.
pinData	A PIN DATA object is returned containing the KSN, and Encrypted PIN Block.

4.8 deviceMessageUserSelection

This delegate will be called when data is returned from **requestResponse**.

Parameter	Description
operationStatus	Status of the selection.
keyValue	Value containing the Key that was pressed by the user.

4.9 deviceMessageDisplayMessageDone

This delegate will be called when data is returned from **setDisplayMessage**.

Parameter	Description
operationStatus	A value is returned containing the status of the Display Message.

4.10 onBypassPINCommandComplete

This delegate will be called when data is returned from **requestBypassPINCommand**.

Parameter	Description
data	A String value is returned containing the Bypass PIN data received from the device

4.11 onSendCommandTimeOutComplete

This delegate will be called when data is returned from **sendSpecialCommand**.

Parameter	Description
data	A String value is returned containing data (if any) from the Time

4.12 deviceMessageDeviceConfiguration

This delegate will be called when data is returned from **requestDeviceConfiguration**.

Parameter	Description
configData	Data array value is returned containing the Device Configuration information from the device.

4.13 deviceMessageARQCRequest

This delegate will be called when data is returned from **requestSmartCard**.

Parameter	Description
ARQCData	Byte array value containing the contents of the ARQC Request received fom the device.

4.14 deviceMessageEMVData

This delegate will be called when data is returned from **sendAcquirerResponse**.

Parameter	Description
emvData	Byte array value containing the EMV transaction data.

4.15 onRequestPowerUpResetICCAPDUComplete

This delegate will be called when data is returned from Error! Reference source not found..

Parameter	Description
data	A String value is returned containing the ICC APDU data (if any) after the device has completed the Reset/Power Up.

4.16 onRequestChallengeAndSessionKeyRequestComplete

This delegate will be called when data is returned from **requestChallengeAndSessionForInformation**.

Parameter	Description
data	A String value is returned containing the Challenge and Session Key from the device

4.17 onCardRequestComplete

This delegate will be called when data is returned from **requestCard**.

Parameter	Description
statusCode	An unsigned char value containing the status code of the command

Parameter	Description
CARD_DATA_INFO	A CARD DATA INFO object is returned containing all of the necessary card data

4.18 deviceMessageCardStatus

This delegate will be called when data is returned from **requestCard**.

Parameter	Description
operationStatus	A value is returned for the operation status.
cardStatus	A value is returned for the card status.

4.19 deviceMessageEMVTags

This delegate will be called when data is returned from **requestGetEMVTag**.

Parameter	Description
tagData	Byte array value is returned containing the EMV Tag data.

4.20 onRequestSmartCardComplete

This delegate will be called after **sendAcquirerResponse** is completed.

Parameter	Description
statusCode	An unsigned char value containing the status code of the command
data	A String value is returned containing the Smart Card batch data

4.21 onSendAcquirerResponseComplete

This delegate will be called when data is returned from **sendAcquirerResponse**.

Parameter	Description
statusCode	An unsigned char value containing the status code of the command
data	A String value is returned containing the Acquirer Response data

4.22 deviceMessageUserDataEntry

This delegate will be called when data is returned from **requestUserDataEntry**.

Parameter	Description
operationStatus	Value containing the status code of the command.
USER_DATA_ENTRY	A USER DATA ENTRY object is returned containing the Encrypted Data Block and the MSR KSN.

4.23 deviceMessageKernelInformation

This delegate will be called when data is returned from requestKeyInformation.

Parameter	Description
kernelID	Value containing the kernel ID.
kernelInformation	Data array containing the value for the kernel ID.

4.24 deviceNotPaired

This delegate will be called when device is not paired when sending command.

4.25 deviceMessageACKStatus

This delegate will be called when data is returned from various requests.

Parameter	Description
status	Status of the operation.
commandID	Byte value for the commandID.

4.26 deviceMessageClearTextUserEntry

This delegate will be called when data is returned from **requestClearTextUserDataEntry**.

Parameter	Description
operatiostatus	Status of the operation.
clearTextData	Byte array value containing the clear text user data.

4.27 deviceMesageDeviceInformation

This delegate will be called when data is returned from **requestDeviceInformation**.

Parameter	Description
type	Type of device information.
data	Byte array value containing the device information data.

4.28 deviceMessageUserSignature

This delegate will be called when data is returned from **requestSignature**.

Parameter	Description
status	Status of the operation.
signature	Byte array value containing the signature data.

4.29 deviceMessageDeviceStatus

This delegate will be called when data is returned from requestDeviceStatus.

Parameter	Description
deviceStatus	Status of the device.

4.30 deviceMessageKeyInformation

This delegate will be called when data is returned from requestDeviceStatus.

Parameter	Description
keyID	Value containing the ID of the key.
keyStatus	Value indicating the status of key.
keyInfo	Byte array value containing the key information.

4.31 deviceMessageBINTableData

This delegate will be called when data is returned from **requestBINTableData**.

Parameter	Description
binData	Byte array value containing the BIN table data.

4.32 deviceMessageTipOrCashback

This delegate will be called when data is returned from **requestTipOrCashback**.

Parameter	Description
status	Status of the operation.
mode	Mode of either Tip or Cashback.
amount	Amount from requestTipOrCashback request.
tax	Tax amount from requestTipOrCashback request.
taxRate	Tax rate percent from requestTipOrCashback request.
tipOrCashBack	Tip or Cashback amount depending on the mode specified.
reserve	Reserved

4.33 deviceMessageSelectedMenuItem

This delegate will be called when data is returned from **getSelectedMenuItem**.

Parameter	Description
status	Status of the menu selection.
mode	Mode of the menu selection.
index	Index of the selected menu item.
reserve	Reserved

4.34 onConnectionStateChanged

This delegate will be called when there is change in the connection state.

Parameter	Description
connectionType	Value of the connection type.
state	Value of the connection state.

4.35 onCertificateError

This delegate will be called when there is an error returned from **loadClientCertificate**.

Parameter	Description
error	Value containing the error code.

Appendix A Code Examples

A.1 Instantiating the MTPPSCRA Library

```
// This can be performed within viewDidLoad in the class needed.
// or the object can be created once within the Application’s Delegate
// class and used throughout the entire project.
MTPPSCRA *mtPPSCRA = [MTPPSCRA shareInstance];

//**** set the MTPPSCRA Delegate to self
mtPPSCRA.delegate2 = self;

//**** set the MTPPSCRA Library
 [mtPPSCRA setMTPPSCRALibrary];
```

A.2 Open Device

```
MTPPSCRA *mtPPSCRA = [MTPPSCRA shareInstance];

[mtPPSCRA setConnectionType:BLEEMV];

// load previously connected Peripheral's UUID (if any)
NSString *uuidString = [self loadPeripheral];

[mtPPSCRA setDeviceUUIDString:uuidString];

 [self openDevice];
```

A.3 Close Device

```
MTPPSCRA *mtPPSCRA = [MTPPSCRA shareInstance];

 [self closeDevice];
```