

DynaMAX, eDynamo, aDynamo,
uDynamo, iDynamo 5 Gen II,
iDynamo 5 Gen III, iDynamo 6,
kDynamo, sDynamo, tDynamo

**Secure Card Reader Authenticators
SDK Programmer's Manual (iOS)**

June 2025

Document Number:
D99875568-133

REGISTERED TO ISO 9001:2015

Copyright © 2006 – 2025 MagTek, Inc.
Printed in the United States of America

INFORMATION IN THIS PUBLICATION IS SUBJECT TO CHANGE WITHOUT NOTICE AND MAY CONTAIN TECHNICAL INACCURACIES OR GRAPHICAL DISCREPANCIES. CHANGES OR IMPROVEMENTS MADE TO THIS PRODUCT WILL BE UPDATED IN THE NEXT PUBLICATION RELEASE. NO PART OF THIS DOCUMENT MAY BE REPRODUCED OR TRANSMITTED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, FOR ANY PURPOSE, WITHOUT THE EXPRESS WRITTEN PERMISSION OF MAGTEK, INC.

MagTek®, MagnePrint®, and MagneSafe® are registered trademarks of MagTek, Inc.
Magensa™ is a trademark of MagTek, Inc.
IPAD® is a trademark of MagTek, Inc.

AAMVA™ is a trademark of AAMVA.
American Express® and EXPRESSPAY FROM AMERICAN EXPRESS® are registered trademarks of American Express Marketing & Development Corp.
D-PAYMENT APPLICATION SPECIFICATION® is a registered trademark to Discover Financial Services CORPORATION
MasterCard® is a registered trademark and PayPass™ and Tap & Go™ are trademarks of MasterCard International Incorporated.
Visa® and Visa payWave® are registered trademarks of Visa International Service Association.

ANSI®, the ANSI logo, and numerous other identifiers containing “ANSI” are registered trademarks, service marks, and accreditation marks of the American National Standards Institute (ANSI).
ISO® is a registered trademark of the International Organization for Standardization.
UL™ and the UL logo are trademarks of UL LLC.
PCI Security Standards Council® is a registered trademark of the PCI Security Standards Council, LLC.
EMV® is a registered trademark in the U.S. and other countries and an unregistered trademark elsewhere. The EMV trademark is owned by EMVCo, LLC. The Contactless Indicator mark, consisting of four graduating arcs, is a trademark owned by and used with permission of EMVCo, LLC.
The *Bluetooth*® word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by MagTek is under license.

Apple Pay®, iPhone®, iPod®, Mac®, and OS X® are registered trademarks of Apple Inc., registered in the U.S. and other countries. iPad™ is a trademark of Apple, Inc. App StoreSM is a service mark of Apple Inc., registered in the U.S. and other countries. IOS is a trademark or registered trademark of Cisco in the U.S. and other countries and is used by Apple Inc. under license.
Microsoft®, Windows®, and .NET® are registered trademarks of Microsoft Corporation.

All other system names and product names are the property of their respective owners.

Table 0-1 – Revisions

Rev Number	Date	Notes
1.01	2011 Dec 22	Initial Release
1.02	2011 Aug 02	Remove Audio reader
1.03	2012 Feb 29	Added Functionality
1.04	2012 Mar 26	Added getBatteryLevel
1.05	2012 Apr 19	Added getSDKVersion & getOperationStatus
1.06	2012 May 01	Made iDynamo-specific
20	2015 Jan 30	Added DynaMAX; reformat; added introduction and how to set up; included aDynamo and uDynamo; general cleanup and clarifying detail. Updated the MTSCRATransactionData enum. Updated the MTSCRADeviceType enum. Added delegate methods: onDataReceived, cardSwipeDidStart, cardSwipeDidGetTransError, onDeviceConnectionDidChange, bleReaderConnected, bleReaderDidDiscoverPeripheral, bleReaderStateUpdated
30	2015 Nov 19	Added support for eDynamo Added new functions for eDynamo: onTransactionStatus onDisplayMessageRequest onUserSelectionRequest onARQCReceived onTransactionResult onEMVCommandResult Update return code for EMV function Return Value: 0 = Success 9 = Error 15 = Busy
40	2016 Jul 6	Added DynaPro format for EMV transaction messages. Added getCardPAN function. Added deviceNotPaired delegate.
50	2017 Feb 9	Updated the tested operating systems.
60	2017 Oct 3	Added support for kDynamo, sDynamo, tDynamo.
70	2018 Aug 6	Added device type kDynamo and tDynamo to setDeviceType().
80	2018 Sep 14	Added onDeviceResponse delegate. Updated startTransaction to support Quick Chip mode.

Rev Number	Date	Notes
90	2019 Sep 16	Updated events for the delegate onTransactionStatus(), and result codes for the delegate onEMVCommandResult(). Updated the function startTransaction(): cardType and transactionType.
91	2020 Jul 16	Update supported iOS versions; Add Bluetooth permission information for iOS 13 in section 3; Add startScanningForPeripherals and stopScanningForPeripherals and add details about the Bluetooth LE connection sequence; Remove Appendix E, F and in some sections, replaced copy-pasted text from other documents with cross references to command manual for the device; Misc. clarifications and corrections.
92	2020 Aug 27	Added 1 new function getBluetoothRSSI, and 2 callback: bleReaderDidDisconnected and debugInfoCallback under sections: 4.51, 5.18, 5.19
100	2021 Aug 18	Added new functions sendCommandSync() and sendExtendedCommandSync()
110	2021 Dec 13	Added setTimeout(), setTimeFrame(), and enableDebugPrint(). Added enums to current SDK.
120	January 31, 2023	Added information table for extended command set date and time, and for MSR head power. Added setup for Swift project. Added more details for setting up Bluetooth LE. Added transaction diagrams.
130	June 20, 2023	Added 0 for transaction timeout of startTransaction() at section 4.55.
131	March 11, 2024	Added support for iDynamo5 Gen III under section Appendix C and Appendix D.2. Added functions starting at section 4.65: updateFirmware(), sendNFCCCommand(), sendClassicNFCCCommand(), sendDESFireNFCCCommand(), sendNFCCCommandSync(), sendNFCCCommandAsync()
132	April 18, 2024	Added support for iPhone 15 and XCode 15.3 at section 1.4. Updated support iOS 13 and above at section 1.5.
133	June 11, 2025	Added SCDE properties to MTCardData (section 6.1).

LIMITED WARRANTY

MagTek warrants that the products sold pursuant to this Agreement will perform in accordance with MagTek's published specifications. This warranty shall be provided only for a period of one year from the date of the shipment of the product from MagTek (the "Warranty Period"). This warranty shall apply only to the "Buyer" (the original purchaser, unless that entity resells the product as authorized by MagTek, in which event this warranty shall apply only to the first repurchaser).

During the Warranty Period, should this product fail to conform to MagTek's specifications, MagTek will, at its option, repair or replace this product at no additional charge except as set forth below. Repair parts and replacement products will be furnished on an exchange basis and will be either reconditioned or new. All replaced parts and products become the property of MagTek. This limited warranty does not include service to repair damage to the product resulting from accident, disaster, unreasonable use, misuse, abuse, negligence, or modification of the product not authorized by MagTek. MagTek reserves the right to examine the alleged defective goods to determine whether the warranty is applicable.

Without limiting the generality of the foregoing, MagTek specifically disclaims any liability or warranty for goods resold in other than MagTek's original packages, and for goods modified, altered, or treated without authorization by MagTek.

Service may be obtained by delivering the product during the warranty period to MagTek (1710 Apollo Court, Seal Beach, CA 90740). If this product is delivered by mail or by an equivalent shipping carrier, the customer agrees to insure the product or assume the risk of loss or damage in transit, to prepay shipping charges to the warranty service location, and to use the original shipping container or equivalent. MagTek will return the product, prepaid, via a three (3) day shipping service. A Return Material Authorization ("RMA") number must accompany all returns. Buyers may obtain an RMA number by contacting MagTek Support Services at (888) 624-8350.

EACH BUYER UNDERSTANDS THAT THIS MAGTEK PRODUCT IS OFFERED AS-IS. MAGTEK MAKES NO OTHER WARRANTY, EXPRESS OR IMPLIED, AND MAGTEK DISCLAIMS ANY WARRANTY OF ANY OTHER KIND, INCLUDING ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

IF THIS PRODUCT DOES NOT CONFORM TO MAGTEK'S SPECIFICATIONS, THE SOLE REMEDY SHALL BE REPAIR OR REPLACEMENT AS PROVIDED ABOVE. MAGTEK'S LIABILITY, IF ANY, SHALL IN NO EVENT EXCEED THE TOTAL AMOUNT PAID TO MAGTEK UNDER THIS AGREEMENT. IN NO EVENT WILL MAGTEK BE LIABLE TO THE BUYER FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF, OR INABILITY TO USE, SUCH PRODUCT, EVEN IF MAGTEK HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

LIMITATION ON LIABILITY

EXCEPT AS PROVIDED IN THE SECTIONS RELATING TO MAGTEK'S LIMITED WARRANTY, MAGTEK'S LIABILITY UNDER THIS AGREEMENT IS LIMITED TO THE CONTRACT PRICE OF THIS PRODUCT.

MAGTEK MAKES NO OTHER WARRANTIES WITH RESPECT TO THE PRODUCT, EXPRESSED OR IMPLIED, EXCEPT AS MAY BE STATED IN THIS AGREEMENT, AND MAGTEK DISCLAIMS ANY IMPLIED WARRANTY, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

MAGTEK SHALL NOT BE LIABLE FOR CONTINGENT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES TO PERSONS OR PROPERTY. MAGTEK FURTHER LIMITS ITS LIABILITY OF ANY KIND WITH RESPECT TO THE PRODUCT, INCLUDING NEGLIGENCE ON ITS PART, TO THE CONTRACT PRICE FOR THE GOODS.

MAGTEK'S SOLE LIABILITY AND BUYER'S EXCLUSIVE REMEDIES ARE STATED IN THIS SECTION AND IN THE SECTION RELATING TO MAGTEK'S LIMITED WARRANTY.

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.

DynaMAX, eDynamo, aDynamo, uDynamo, iDynamo 5 Gen II, iDynamo 5 Gen III, iDynamo 6, kDynamo, sDynamo, tDynamo | Secure Card Reader Authenticators | SDK Programmer's Manual (iOS)

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.

DEMO SOFTWARE / SAMPLE CODE: Unless otherwise stated, all demo software and sample code are to be used by Licensee for demonstration purposes only and MAY NOT BE incorporated into any production or live environment. The PIN Pad sample implementation is for software PIN Pad test purposes only and is not PCI compliant. To meet PCI compliance in production or live environments, a third-party PCI compliant component (hardware or software-based) must be used.

Table of Contents

Limited Warranty	5
SOFTWARE LICENSE AGREEMENT	7
Table of Contents	9
1 Introduction	13
1.1 About MTSCRADemo	13
1.2 Nomenclature	13
1.3 SDK Contents	13
1.4 System Requirements	14
1.5 Interfaces for Operating Systems	14
2 How to Set Up the MTSCRA SDK	15
2.1 Setup for XCode Project	15
2.2 Setup for Swift Project.....	16
3 Important Information About Bluetooth LE.....	19
4 MTSCRA Functions	21
4.1 getSDKVersion	21
4.2 startScanningForPeripherals.....	21
4.3 stopScanningForPeripherals	21
4.4 setAddress.....	22
4.5 openDevice.....	22
4.6 closeDevice	22
4.7 isDeviceConnected.....	22
4.8 isDeviceOpened.....	23
4.9 sendCommandToDevice	23
4.10 getResponseData.....	23
4.11 clearBuffers.....	24
4.12 listenForEvents	24
4.13 getMaskedTracks [iDynamo/uDynamo Only].....	24
4.14 getTrack1Masked	24
4.15 getTrack2Masked	25
4.16 getTrack3Masked	25
4.17 getCardPAN.....	25
4.18 getTrack1	25
4.19 getTrack2	25
4.20 getTrack3	25
4.21 getMagnePrint	26
4.22 getMagnePrintStatus [iDynamo Only]	26
4.23 getDeviceSerial.....	26
4.24 getMagTekDeviceSerial	26

4.25	getSessionID [iDynamo/uDynamo Only].....	26
4.26	getKSN	27
4.27	getDeviceName	27
4.28	getDeviceType.....	27
4.29	setDeviceType.....	27
4.30	getDeviceCaps	28
4.31	getCapMSR	28
4.32	getCapMagStripeEncryption.....	28
4.33	getCapTracks	29
4.34	getCardExpDate.....	29
4.35	getCardLast4	29
4.36	getCardIIN.....	29
4.37	getCardName.....	29
4.38	getCardPANLength	29
4.39	getCardServiceCode	30
4.40	getFirmware	30
4.41	getTrackDecodeStatus	30
4.42	getBatteryLevel.....	31
4.43	getDevicePartNumber	31
4.44	getCardStatus	31
4.45	getTagValue [aDynamo/uDynamo Only].....	31
4.46	getDeviceStatus	31
4.47	getOperationStatus.....	31
4.48	getTLVVersion	32
4.49	setDeviceProtocolString [iDynamo Only].....	32
4.50	getResponseTypes	32
4.51	getBluetoothRSSI	32
4.52	setUUIDString [DynaMAX/eDynamo Only]	32
4.53	getConnectedPeripheral [DynaMAX/eDynamo Only].....	33
4.54	getDiscoveredPeripherals [DynaMAX/eDynamo Only]	33
4.55	startTransaction (EMV Device Only)	33
4.56	setUserSelectionResult (EMV Only)	36
4.57	cancelTransaction	36
4.58	setAcquirerResponse (EMV Device Only)	36
4.59	sendExtendedCommand (EMV Device and iDynamo 5 Gen III Only).....	37
4.60	sendCommandSync.....	37
4.61	sendExtendedCommandSync (EMV Device Only)	38
4.62	setTimeout	38
4.63	setTimeFrame.....	38
4.64	enableDebugPrint	38
4.65	updateFirmware	39

4.66	sendNFCCCommand	39
4.67	sendClassicNFCCCommand.....	40
4.68	sendDESFireNFCCCommand.....	41
4.69	sendNFCCCommandSync.....	42
4.70	sendNFCCCommandAsync.....	43
5	MTSCRA Delegate Methods	44
5.1	trackDataReadyNotification	44
5.2	devConnectionNotification.....	44
5.3	onDataReceived	44
5.4	cardSwipeDidStart	44
5.5	cardSwipeDidGetTransError	44
5.6	onDeviceConnectionDidChange.....	44
5.7	bleReaderConnected	44
5.8	bleReaderDidDisconnected	45
5.9	bleReaderDidDiscoverPeripheral.....	45
5.10	bleReaderStateUpdated.....	45
5.11	onDeviceResponse.....	45
5.12	onDeviceError	45
5.13	onTransactionStatus (EMV Device Only).....	45
5.14	onDisplayMessageRequest (EMV Device Only)	45
5.15	onUserSelectionRequest (EMV Device Only).....	45
5.16	onARQCReceived (EMV Device Only)	46
5.17	onTransactionResult (EMV Device Only).....	46
5.18	onEMVCommandResult (EMV Device Only)	46
5.19	onDeviceExtendedResponseReceived.....	46
5.20	deviceNotPaired	46
5.21	didGetRSSI	46
5.22	debugInfoCallback.....	47
6	MTSCRA Interfaces.....	48
6.1	MTCardData	48
Appendix A	Troubleshooting.....	51
A.1	Set Date and Time Extended Command 0x030C	51
A.2	Set MSR Head Always On 0x58	52
Appendix B	Code Examples	53
B.1	Open Device	53
B.2	Close Device.....	53
B.3	Get Track Data from Reader	54
B.4	Get Connection Status of Reader	54
B.5	Diagrams	55
B.5.1	MSR Card Swipe.....	55

B.5.2	EMV Transaction (Quick Chip).....	57
B.5.3	EMV Transaction (Full).....	59
Appendix C	Supported Devices.....	61
Appendix D	Enums.....	63
D.1	Error Codes.....	63
D.2	MTSCRADeviceType.....	63
D.3	MTSCRATransactionStatus.....	63
D.4	MTSCRATransactionEvent.....	63
D.5	MTSCRATransactionData.....	63
D.6	MTSCRACapabilities.....	64
D.7	ConnectionTypes.....	64
D.8	DebugDomain.....	64
D.9	MTSCRABLEState.....	65

1 Introduction

This document provides instructions for software developers who want to create custom software solutions that communicate with DynaMAX, eDynamo, aDynamo, iDynamo, uDynamo, kDynamo, sDynamo, tDynamo, iDynamo 5 Gen II, iDynamo 5 Gen III, or iDynamo 6 connected to an iOS host via audio connector or Bluetooth LE. It is part of a larger library of documents which includes:

- *D99875475 MagneSafe V5 Programmer's Reference (Commands)*
- *D998200175 DynaMAX Programmer's Manual (Commands)*
- *D998200115 eDynamo Programmer's Manual (Command)*
- *D998200309 iDynamo 5 (Gen II) Programmer's Manual (Commands)*
- *D998200587 iDynamo 5 (Gen III) Programmer's Manual (Commands)*
- *D998200230 kDynamo Programmer's Manual (Commands)*
- *D998200226 tDynamo Programmer's Manual (Commands)*
- *D998200324 iDynamo 6 Programmer's Manual (Command)*

1.1 About MTSCRADemo

The MTSCRADemo software, available from MagTek, provides demonstration source code and a reusable MTSCRA library that provides developers of custom iOS software solutions with an easy-to-use interface. Developers can include the MTSCRA library in custom branded software which can be distributed to customers or distributed 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 MSR device that receives and responds to the command set specified in this document; in this case, DynaMAX, eDynamo, aDynamo, iDynamo, uDynamo, kDynamo, sDynamo, tDynamo, or iDynamo 6.
- **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 device. When “host” must be used differently, it is qualified as something specific, such as “USB host.”

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

1.3 SDK Contents

File Name	Description
MTSCRA.h	Header file for the MTSCRA SDK
libMTSCRA.a	Library binary for the MTSCRA SDK
MTSCRADemo Folder	Sample code and projects

1.4 System Requirements

Tested devices:

- iPhone 7, 7 Plus, 8, 8 Plus, X, XS, XS Max, 11, 12, 13, 14, 15 series, iPhone 12 and 13 mini
- iPad 6th, 7th, 8th, 9th, 10th gen, iPad Air, iPad Air 2/3/4/5, iPad Pro, iPad Pro 2nd, 3rd, 4th, and 5th gen
- iPad Mini 2, iPad Mini 3, iPad Mini 4, iPad Mini 5, iPad Mini 6

Tested operating systems: iOS 13 and above

Build Platforms: XCode 15.3 and above

1.5 Interfaces for Operating Systems

The following table matches the device interface to operating system.

Device	Interface	Operating System
eDynamo	Bluetooth LE 4.0	iOS 13 and above
aDynamo	Audio	iOS 13 and above
iDynamo	30-pin	iOS 13 and above
	Lightning	iOS 13 and above
uDynamo	Audio	iOS 13 and above
DynaMAX	Bluetooth LE 4.0	iOS 13 and above
kDynamo	Lightning	iOS 13 and above
sDynamo	Lightning	iOS 13 and above
tDynamo	Bluetooth LE 4.0	iOS 13 and above
iDynamo 5 Gen II	Lightning	iOS 13 and above
iDynamo 5 Gen III	USB-C (iAP2)	iOS 13 and above
iDynamo 6	Lightning	iOS 13 and above

2 How to Set Up the MTSCRA SDK

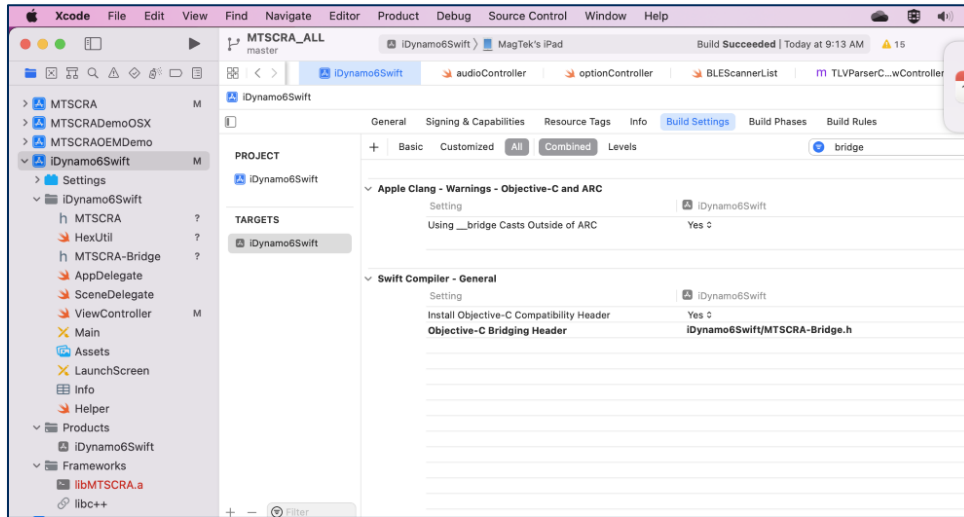
2.1 Setup for XCode Project

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

- 1) Download the MTSCRA Demo app from MagTek.com.
- 2) Open your custom software project in XCode.
- 3) Open the MTSCRA Demo app folder in Finder.
- 4) Open the `Lib` subfolder.
- 5) Include the following files in your custom software project within XCode:
 - a) `libMTSCRA.a`
 - b) `MTSCRA.h`
- 6) Ensure the library search paths are set up correctly.
- 7) If you are writing an app that will connect to Bluetooth LE devices, see the additional steps section **3 Important Information About Bluetooth LE**.
- 8) Clean, build, and run your custom software project to make sure the library imported correctly.
- 9) In your custom software, create an instance of `MTSCRA`. For examples, including how to register delegate functions in your app for callbacks, see the source code included with the MTSCRA Demo app and / or Appendix C Code Examples .
- 10) Begin using the features provided by the `MTSCRA` object's methods. For details about these methods, see section **4 MTSCRA Functions**

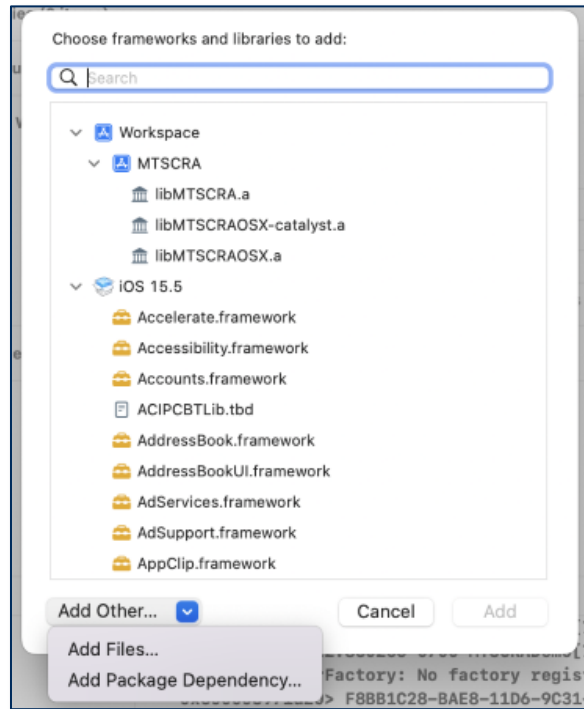
2.2 Setup for Swift Project

- 1) Add the following files to project, and setup the bridge header folder in build settings.
 - a) MTSCRA.H
 - b) MTSCRA-Bridge.h

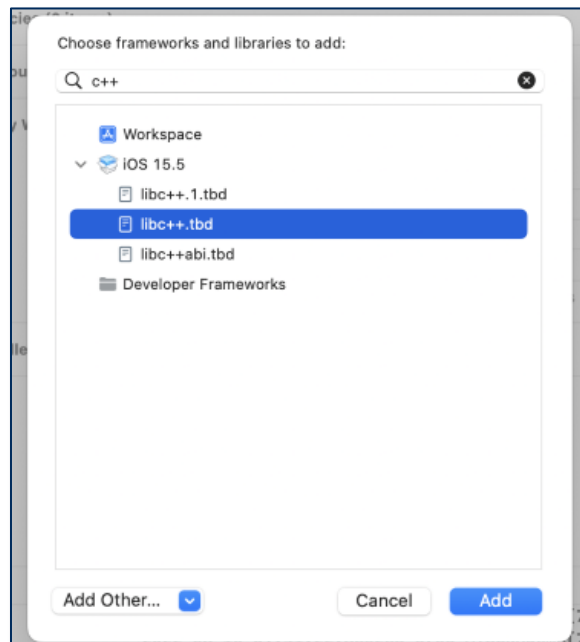


2 - How to Set Up the MTSCRA SDK

- 2) Add `libMTSCRA.a`, in build phases->Link Binary with libraries, click '+', then select Add Files and browse to `libMTSCRA.a`.

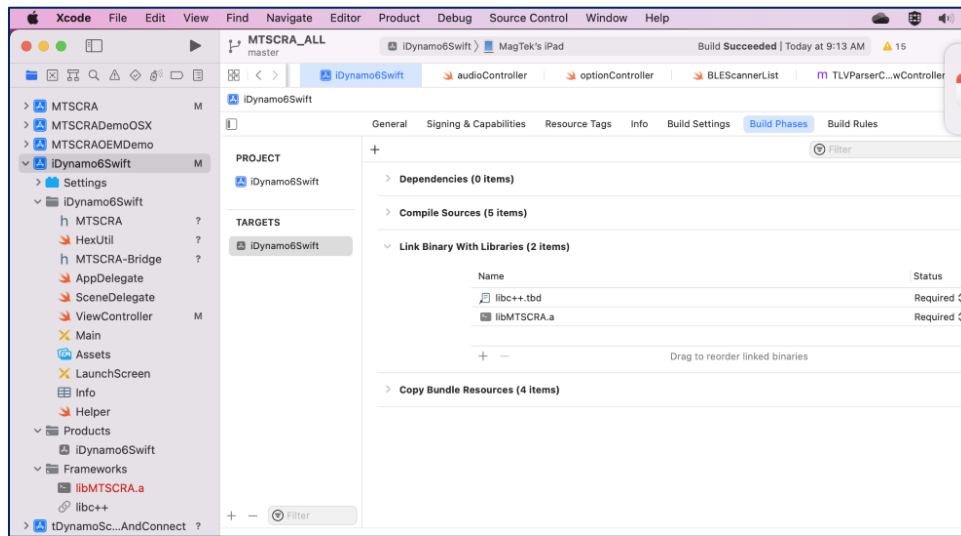


- 3) Add `libc++.tbd` in build phases->Link Binary with libraries, click '+', then type "c++" in the filter area and pick `libc++.tbd`.

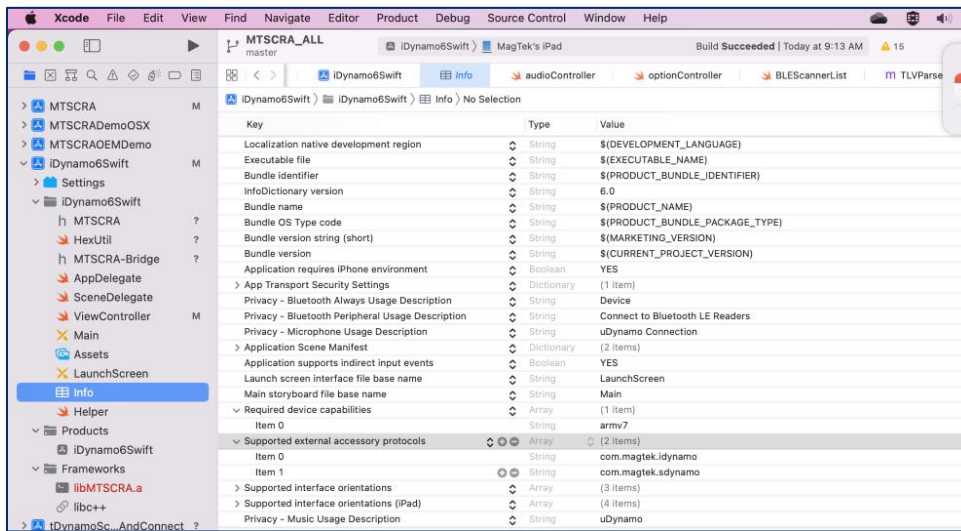


2 - How to Set Up the MTSCRA SDK

4) After adding those two libraries, this will be in the project.

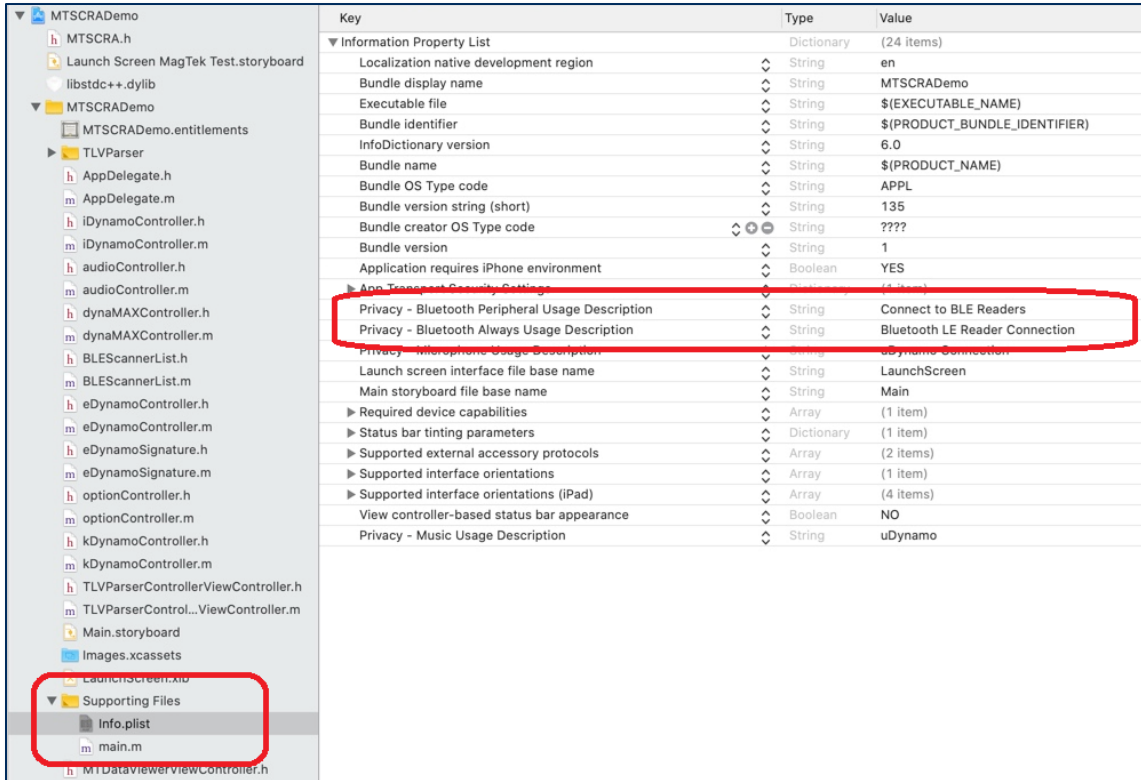


5) Add the external accessory protocols into info file. In the screen shot, it adds iDynamo5 and iDynamo6 with default protocols. Add your custom protocol if different.



3 Important Information About Bluetooth LE

- 1) When calling functions `startScanningForPeripherals` or `openDevice`, the application should make sure `bleReaderStateUpdated` has received a device status and that the most recent status was **OK**, otherwise the device will not be able to connect, and iOS will not throw any error if Bluetooth is not ready.
- 2) Starting in iOS 13, app projects must specify the Privacy Usage Description for Bluetooth by including `NSBluetoothAlwaysUsageDescription` in the `Info.plist` file. Accessing Core Bluetooth without the usage descriptions will cause a runtime crash. For backward compatibility with older versions of iOS, define `NSBluetoothPeripheralUsageDescription` as well.



- 3) Make sure device is in pairing mode. If not, press and hold the button for 3 seconds until the blue light blinks, and then release the button.
- 4) Set device type and connection type.

```
self.lib.setConnectionType(UInt(BLE_EMV))
self.lib.setDeviceType(UInt32(MAGTEKEDYNAMO))
```

- 5) Wait for callback `bleReaderStateUpdated()`, and then call `startScanningForPeripheral()`.

```
func bleReaderStateUpdated(_ state: MTSCRABLEState) {
    print("readerStateUpdated - ", state)

    if (state == 0) {
        // dispatch in main queue is very important
    }
}
```

3 - Important Information About Bluetooth LE

```
DispatchQueue.main.asyncAfter(deadline: .now() + 0.1,
execute: {
    self.lib.startScanningForPeripherals()
})
}
```

6) Wait for `onDeviceList()` callback. Here you can call `getDiscoveredPeripherals()`.

```
func onDeviceList(_ instance: Any!, connectionType: UInt, deviceList:
[Any]!) {
    let devices = deviceList as! [MTDeviceInfo]
    devList.removeAll()

    let peripherals = self.lib.getDiscoveredPeripherals()
    print(peripherals as Any)
    ...
}
```

7) Select the device in the list.

8) Open the device you want to connect.

```
lib.stopScanningForPeripherals()

lib.setAddress(devList[selected].address)
lib.openDevice()
```

9) Library will pair the device. After inputting passcode, device is ready to use.

4 MTSCRA Functions

To develop an iOS app using the MTSCRA SDK, follow the setup steps in section 2 **How to Set Up the MTSCRA SDK**, then create an instance of the MTSCRA object in your software project, then call the functions described in this chapter to communicate with the device. For sample code that demonstrates how to use these functions, see the contents of the **MTSCRADemo** folder included with the SDK.

Generally, these functions will run in one of two modes:

- **Asynchronous** functions will return data using the event handlers (callback functions) defined in section 5 **MTSCRA Delegate Methods**.
- **Synchronous** functions will return requested data immediately in the function's return value. If the requested data is not available immediately, synchronous calls will generally block until a specified wait time has elapsed.

Most calls that wait for input from the user will run in the asynchronous mode.

4.1 getSDKVersion

This function retrieves the SDK revision number.

```
(NSString *) getSDKVersion
```

Parameters: None

Return Value: String containing the SDK revision number.

4.2 startScanningForPeripherals

This function retrieves a list of available Bluetooth LE devices. After calling this function to locate the device you wish to connect to, use **setAddress** to tell the library which device you want to connect to. Use **stopScanningForPeripherals** to stop the scan.

```
(void)startScanningForPeripherals
```

Parameters: None

Return Value: An array of peripherals

4.3 stopScanningForPeripherals

This function stops the scanning of available Bluetooth LE devices.

```
(void)stopScanningForPeripherals
```

Parameters: None

Return Value: None

4.4 setAddress

This function sets device's address for Bluetooth LE devices.

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

Parameters:

Parameter	Description
address	Address of the Bluetooth LE device to communicate with

Return Value: None

4.5 openDevice

This function opens a connection to the device.

To use this function to connect to a Bluetooth LE device, the app should follow these steps:

- 1) Call **startScanningForPeripherals** to find the device you want to connect to.
- 2) Call **setAddress** to tell the library which device you want to connect to.
- 3) Make sure your app's **bleReaderStateUpdated** function most recent status is was **OK**.
- 4) Call **openDevice**.

After calling this function, call **isDeviceOpened** to make sure the device was successfully opened.

```
(BOOL) openDevice
```

Parameters: None

Return Value:

- YES = Success
- NO = Error

4.6 closeDevice

This function closes the connection to the currently opened device. After calling this function, call **isDeviceOpened** to make sure the device was successfully closed.

```
(BOOL) closeDevice
```

Parameters: None

Return Value:

- YES = Success
- NO = Error

4.7 isDeviceConnected

This function reports whether any compatible devices are connected to the host.

```
(BOOL) isDeviceConnected
```

Parameters: None

Return Value:

- YES = host is connected to a device
- NO = host is not connected to a device

4.8 isDeviceOpened

This function retrieves device opened status, which changes on successful completion of a call to **startScanningForPeripherals** or **closeDevice**.

```
(BOOL) isDeviceOpened
```

Parameters: None

Return Value:

- YES = Device is opened
- NO = Device is not opened

4.9 sendCommandToDevice

This function sends a direct command to device. See *D99875475 MagneSafe V5 Programmer's Reference (Commands)* for details about available commands and syntax.

```
(int) sendCommandToDevice:(NSString *)pData
```

Parameters:

Parameter	Description
pData	Command to send to the device. For example, pass command string "C10206C20503C30100" to call the Discovery command.

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

4.10 getResponseData

This function retrieves card data from a string separated by '|' after a cardholder swipes a card. The host software should call it in response to the **trackDataReadyNotification** callback.

```
(NSString *) getResponseData
```

Parameters: None

Return Value:

A null terminated hex string for Card Data, Field separated by '|'.NULL value for failed.

Fields:

Device ID, Device Serial Number, Card Swipe Status, CardEncode Type, Track 1 Decode Status, Track 2 Decode Status, Track 3 Decode Status, MagnePrint Status, Track 1 Length, Track 2 Length, Track 3 Length, Masked Track 1 Length, Masked Track 2 Length, Masked Track 3 Length, MagnePrint Length, Card Data, Masked Card Data, DUKPT Session ID, DUKPT Key Serial Number, First Name, Last Name, PAN, Month, Year, Track 1 Data, Track 2 Data, Track 3 Data, Masked Track 1 Data, Masked Track 2 Data, Masked Track 3 Data, MagnePrint Data

4.11 clearBuffers

This function clears the SDK library's local cache of card swipe data.

```
(void) clearBuffers
```

Parameters: None

Return Value: None

4.12 listenForEvents

This function sets a callback function to notify when the device has card data to send to the host or when the device state changes. See example in **Open Device** code example.

```
(void) listenForEvents:(UInt32)event
```

Parameters: Event

Parameter	Description
event	Event type. <ul style="list-style-type: none">TRANS_EVENT_OK = Transaction succeeded.TRANS_EVENT_START = Reader started sending data.TRANS_EVENT_ERROR = Reader failed sending data.

Return Value: None

4.13 getMaskedTracks [iDynamo/uDynamo Only]

This function retrieves masked card track data after a cardholder swipes a card. Only available on iDynamo/uDynamo; other devices will return an empty string.

```
(NSString *) getMaskedTracks
```

Parameters: None

Return Value:

Return stored masked track data string. Tracks are delimited with start and end sentinels.

4.14 getTrack1Masked

This function retrieves masked track 1 data after a cardholder swipes a card.

```
(NSString *) getTrack1Masked
```

Parameters: None

Return Value: Return stored masked track1 data string.

4.15 getTrack2Masked

This function retrieves masked track 2 data, if any, after a cardholder swipes a card.

```
(NSString *) getTrack2Masked
```

Parameters: None

Return Value: Return stored masked track2 data string.

4.16 getTrack3Masked

This function retrieves masked track 3 data, if any, after a cardholder swipes a card.

```
(NSString *) getTrack3Masked
```

Parameters: None

Return Value: Return stored masked track3 data string.

4.17 getCardPAN

This function retrieves masked PAN data, if any, after a cardholder swipes a card.

```
(NSString *) getCardPAN
```

Parameters: None

Return Value: Return stored masked PAN data string.

4.18 getTrack1

This function retrieves the card's track 1 data in encrypted format after a cardholder swipes a card.

```
(NSString *) getTrack1
```

Parameters: None

Return Value: String containing encrypted track 1 data.

4.19 getTrack2

This function retrieves the card's track 2 data in encrypted format, if any, after a cardholder swipes a card.

```
(NSString *) getTrack2
```

Parameters: None

Return Value: String containing encrypted track 2 data.

4.20 getTrack3

This function retrieves the card's track 3 data in encrypted format, if any, after a cardholder swipes a card.

```
(NSString *) getTrack3
```

Parameters: None

Return Value: String containing encrypted track 3 data.

4.21 getMagnePrint

This function retrieves the card's encrypted MagnePrint, for readers that support MagnePrint.

```
(NSString *) getMagnePrint
```

Parameters: None

Return Value: String containing the card's encrypted MagnePrint.

4.22 getMagnePrintStatus [iDynamo Only]

This function retrieves the card MagnePrint status. For more information, see **D99875475**. Only available on iDynamo; it will return an empty string in audio reader.

```
(NSString *) getMagnePrintStatus
```

Parameters: None

Return Value:

Return stored MagnePrintStatus string.

4.23 getDeviceSerial

This function retrieves the device serial number.

```
(NSString *) getDeviceSerial
```

Parameters: None

Return Value: String containing the device serial number.

4.24 getMagTekDeviceSerial

This function returns the MagTek serial number of the currently opened device.

```
(NSString *) getMagTekDeviceSerial
```

Parameters: None

Return Value: Return stored serial number created by MagTek.

4.25 getSessionID [iDynamo/uDynamo Only]

This function retrieves the Session ID from the currently opened device, which the host can use to uniquely identify a transaction to prevent replay. Only supported by iDynamo/uDynamo; on other devices this function will return an empty string. For more information, see **D99875475**

```
(NSString *) getSessionID
```

Parameters: None

Return Value: Stored session ID.

4.26 getKSN

This function retrieves the Key Serial Number (KSN) from the device.

```
(NSString *) getKSN
```

Parameters: None

Return Value: String containing the stored key serial number.

4.27 getDeviceName

This function gets the device's product name.

```
(NSString *) getDeviceName
```

Parameters: None

Return Value: String containing the device product name.

4.28 getDeviceType

This function gets the device type.

```
(int) getDeviceType
```

Parameters: None

Return Value: Device Type

4.29 setDeviceType

This function sets the type of device to open. Call this function before calling openDevice.

```
(void) setDeviceType:(UInt32 *)deviceType
```

Parameters:

Device Type:

- MAGTEKAUDIOREADER = Audio readers aDynamo, uDynamo.
- MAGTEKIDYNAMO = iOS 30-pin and Lightning readers iDynamo.
- MAGTEKDYNAMAX = Bluetooth LE reader DynaMAX.
- MAGTEKEDYNAMO = Bluetooth LE reader eDynamo.
- MAGTEKKDYNAMO = Lightning reader kDynamo.
- MAGTEKTDYNAMO = Bluetooth LE reader tDynamo.

Return Value: None

4.30 getDeviceCaps

This function gets the capabilities of the currently opened device.

```
(NSString *) getDeviceCaps
```

Parameters: None

Return Value: Return device capabilities.

- CAP_MASKING = 1,
- CAP_ENCRYPTION=2,
- CAP_CARD_AUTH = 4,
- CAP_DEVICE_AUTH = 8,
- CAP_SESSION_ID = 16,
- CAP_DISCOVERY= 32,

4.31 getCapMSR

This function gets the MSR capability of the device. For more information, see **D99875483** – Track ID Enable Property.

```
(NSString *) getCapMSR
```

Parameters: None

Return Value:

Return MSR Capability bit masking.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Id	0	T ₃	T ₃	T ₂	T ₂	T ₁	T ₁

Id 0 – Decodes standard ISO/ABA cards only

1 – Decodes AAMV and 7-bit cards also

If this flag is set to 0, only tracks that conform to the ISO format allowed for that track will be decoded. If the track cannot be decoded by the ISO method it will be considered to be in error.

T# 00 – Track Disabled

01 – Track Enabled

10 – Track Enabled/Required (Error if blank)

4.32 getCapMagStripeEncryption

This function gets the device’s capability for encrypting track data.

```
(NSString *) getCapMagStripeEncryption
```

Parameters: None

Return Value:

- “1” = Available

- “0” = Unavailable.

4.33 getCapTracks

This function gets information about the device’s tracks capability.

```
(NSString *) getCapTracks
```

Parameters: None

Return Value: A hex string for the track capability. See Track ID Enable Property in **D99875475**.

4.34 getCardExpDate

This function retrieves the card expiration date after a cardholder swipes a card.

```
(NSString *) getCardExpDate
```

Parameters: None

Return Value: String containing the card expiration date

4.35 getCardLast4

This function gets the last 4 digits of the card account number (PAN) after a cardholder swipes a card.

```
(NSString *) getCardLast4
```

Parameters: None

Return Value: String containing the last 4 digits of the PAN

4.36 getCardIIN

This function gets the issuer identification number (IIN) of the card number after a cardholder swipes a card.

```
(NSString *) getCardIIN
```

Parameters: None

Return Value: String containing the IIN

4.37 getCardName

This function gets the cardholder name after a cardholder swipes a card.

```
(NSString *) getCardName
```

Parameters: None

Return Value: String containing the cardholder name, for example, “John Wayne”.

4.38 getCardPANLength

This function gets the length of the PAN after a cardholder swipes a card.

(int) getCardPANLength

Parameters: None

Return Value: Length of card number or PAN

4.39 getCardServiceCode

This function retrieves the card's service code after a cardholder swipes a card.

(NSString *) getCardServiceCode

Parameters: None

Return Value: String containing the card's service code

4.40 getFirmware

This function retrieves the part number and revision of the device's firmware.

(NSString *) getFirmware

Parameters: None

Return Value: String containing firmware part number and revision.

4.41 getTrackDecodeStatus

This function retrieves the track decode status after a cardholder swipes a card.

(NSString *) getTrackDecodeStatus

Parameters: None

Return Value:

Hex string, each 2 digits represent one track's decode status, where the left most 2 digits are for Track 1.

- "00" = Success
- "01" = Error or not decodable
- "02" = No track present

Examples:

- "000000" = Track 1, 2, and 3 success.
- "000100" = Track 1 and 3 success. Track 2 had error.
- "000002" = Track 1 and 2 success. Track 3 not present.

4.42 `getBatteryLevel`

This function retrieves device's battery level percentage between 0% and 100%, if the device has a battery and supports battery level monitoring. To retrieve the most up to date percentage, call this function after a transaction.

```
(long) getBatteryLevel
```

Parameters: None

Return Value: Long value between 0 and 100

4.43 `getDevicePartNumber`

This function returns the currently opened device's part number.

```
(NSString *) getDevicePartNumber
```

Parameters: None

Return Value: String containing the device part number.

4.44 `getCardStatus`

Retrieves the Card Status

```
(NSString *) getCardStatus
```

Parameters: None

Return Value: Card Status, which depends on the device.

4.45 `getTagValue [aDynamo/uDynamo Only]`

This function retrieves individual TLV tag values. Only supported on aDynamo/UDynamo.

```
(NSString *) getTagValue:(UInt32)tag
```

Parameters:

Tag = An `MTSCRATransactionData` type (see section **D.5 MTSCRATransactionData**).

Return Value: String containing the value of the specified tag.

4.46 `getDeviceStatus`

This function gets the status of the currently connected device.

```
(NSString *) getDeviceStatus
```

Parameters: None

Return Value: Return device status of swipe count and battery level.

4.47 `getOperationStatus`

This function gets the status of the current operation.

```
(NSString *) getOperationStatus
```

Parameters: None

Return Value: Operation Status

4.48 getTLVVersion

This function returns the version of the tag-length-value (TLV) format supported by the device.

```
(NSString *) getTLVVersion
```

Parameters: None

Return Value: String containing the firmware TLV version.

4.49 setDeviceProtocolString [iDynamo Only]

This function sets the protocol string the SDK will use to communicate with the device. See example in **Open Device** code example.

```
(void) setDeviceProtocolString:(NSString *)pData
```

Parameters: Protocol String

Return Value: None

4.50 getResponseType

This function gets the response type.

```
(NSString *) getResponseType
```

Parameters: None

Return Value: Response Type

4.51 getBluetoothRSSI

This function will return the signal strength of Bluetooth LE devices.

```
(int) getBluetoothRSSI;
```

4.52 setUUIDString [DynaMAX/eDynamo Only]

This function sets the UUIDString for the Bluetooth LE connection.

```
(void) setUUIDString:(NSString *)uuidString
```

Parameters: UUID String of the device

Return Value: None

4.53 getConnectedPeripheral [DynaMAX/eDynamo Only]

This function gets the current connected peripheral (device).

```
(NSString *) getConnectedPeripheral
```

Parameters: None

Return value: Current connected device

4.54 getDiscoveredPeripherals [DynaMAX/eDynamo Only]

This function gets an array of DynaMAX/eDynamo devices connected to the host.

```
(NSMutableArray *) getDiscoveredPeripherals
```

Parameters: None

Return Value: Array of DynaMAX/eDynamo devices.

4.55 startTransaction (EMV Device Only)

Start EMV Transaction.

The device's system date and time must be set prior to sending this command. Use `sendExentedCommandSync()` to set the date and time.

- Devices without a battery-backed real time clock require the host to set the date and time using Extended Command 0x030C - Set Date and Time every time the device is power cycled or reset.
- Devices that have a battery-backed real time clock (See **Supported Devices**) would typically have the date and time set at the factory.

```
(int) startTransaction:  
(Byte) timeLimit  
cardType: (Byte) cardType  
option: (Byte) option  
amount: (Byte*) amount  
transactionType: (Byte) transactionType  
cashBack: (Byte*) cashBack  
currencyCode: (Byte*) currencyCode  
reportingOption: (Byte) reportingOption
```

Parameters:

Parameter	Description																						
timeLimit	<p>Specifies the maximum time, in seconds, allowed to complete the total transaction. This includes time for the user to insert the card, choose a language, choose an application, and online processing. If this time is exceeded, the transaction will be aborted and an appropriate Transaction Status will be available.</p> <p>Values: 0 = no timeout, infinite. To cancel before a payment method is presented, use cancelTransaction().</p> <p>(1 to 255) seconds. Use 255 for the longest time limit for devices that do not support 0.</p> <table border="1" data-bbox="526 688 961 1092"> <thead> <tr> <th data-bbox="526 688 792 743">Device</th> <th data-bbox="792 688 961 743">Time Limit</th> </tr> </thead> <tbody> <tr> <td data-bbox="526 743 792 785">DynaMAX</td> <td data-bbox="792 743 961 785">n/a</td> </tr> <tr> <td data-bbox="526 785 792 827">aDynamo</td> <td data-bbox="792 785 961 827">n/a</td> </tr> <tr> <td data-bbox="526 827 792 869">cDynamo</td> <td data-bbox="792 827 961 869">n/a</td> </tr> <tr> <td data-bbox="526 869 792 911">iDynamo 5/(GENII)</td> <td data-bbox="792 869 961 911">n/a</td> </tr> <tr> <td data-bbox="526 911 792 953">sDynamo</td> <td data-bbox="792 911 961 953">n/a</td> </tr> <tr> <td data-bbox="526 953 792 995">uDynamo</td> <td data-bbox="792 953 961 995">n/a</td> </tr> <tr> <td data-bbox="526 995 792 1037">eDynamo</td> <td data-bbox="792 995 961 1037">1 to 255</td> </tr> <tr> <td data-bbox="526 1037 792 1079">iDynamo 6</td> <td data-bbox="792 1037 961 1079">0 to 255</td> </tr> <tr> <td data-bbox="526 1079 792 1121">kDynamo</td> <td data-bbox="792 1079 961 1121">1 to 255</td> </tr> <tr> <td data-bbox="526 1121 792 1163">tDynamo</td> <td data-bbox="792 1121 961 1163">0 to 255</td> </tr> </tbody> </table>	Device	Time Limit	DynaMAX	n/a	aDynamo	n/a	cDynamo	n/a	iDynamo 5/(GENII)	n/a	sDynamo	n/a	uDynamo	n/a	eDynamo	1 to 255	iDynamo 6	0 to 255	kDynamo	1 to 255	tDynamo	0 to 255
Device	Time Limit																						
DynaMAX	n/a																						
aDynamo	n/a																						
cDynamo	n/a																						
iDynamo 5/(GENII)	n/a																						
sDynamo	n/a																						
uDynamo	n/a																						
eDynamo	1 to 255																						
iDynamo 6	0 to 255																						
kDynamo	1 to 255																						
tDynamo	0 to 255																						
cardType	<p>Card Type to Read: 0x01 = Magnetic Stripe (as alternative to EMV L2, card swipe causes abort of EMV L2) 0x02 = Contact chip card 0x03 = Magnetic Stripe and Contact chip Card. 0x04 = Contactless chip card 0x05 = Magnetic Stripe and Contactless chip card. 0x06 = Contact chip card and Contactless chip card. 0x07 = Magnetic Stripe, Contact chip card, Contactless chip card.</p> <p>Refer to Appendix C for supported devices.</p>																						

Parameter	Description
option	<p>0x00 = Normal 0x01 = Bypass PIN 0x02 = Force Online 0x04 = Acquirer not available (Note: prevents long timeout on waiting for host approval) (causes “decline” to be generated internally if ARQC is generated)</p> <p>To use Quick Chip mode, set the most significant bit to ‘1’. 0x80 = Quick Chip, Normal 0x81 = Quick Chip, Bypass PIN 0x82 = Quick Chip, Force Online</p> <p>Refer to Appendix C for supported devices.</p>
amount	Amount Authorized (EMV Tag 9F02, format n12, 6 bytes) in hex string For example: “00000000999”, means 9.99 dollars.
transactionType	Valid values: 0x00 = Purchase (listed as “Payment” on ICS) 0x01 = Cash Advance (not supported for this reader) 0x02 or 0x09 = Cash back (0x09 only supported when using contactless) 0x04 = Goods (Purchase) 0x08 = Services (Purchase) 0x10 = International Goods (Purchase) 0x20 = Refund 0x40 = International Cash Advance or Cash Back 0x80 = Domestic Cash Advance or Cash Back
cashBack	Cash back Amount (if non-zero, EMV Tag 9F03, format n12, 6 bytes) in hex string. For example: “000000001000”, means 10.00 dollars.
currencyCode	Transaction Currency Code (EMV Tag 5F2A, format n4, 2 bytes) Sample Valid values: 0x0840 – US Dollar 0x0978 – Euro 0x0826 – UK Pound
reportingOption	This single byte field indicates the level of Transaction Status notifications the host desires to receive during the course of this transaction. 0x00 = Termination Status only (normal termination, card error, timeout, host cancel) 0x01 = Major Status changes (terminations plus card insertions and waiting on user) 0x02 = All Status changes (documents the entire transaction flow)

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

4.56 setUserSelectionResult (EMV Only)

This function sets the user selection result. It should be called after receiving the onUserSelectRequest event which is triggered after the user makes a selection.

```
(int) setUserSelectionResult:(Byte) status selection:(Byte) selection;
```

Parameters:

Parameter	Description
status	Indicates the status of User Selection: 0x00 – User Selection Request completed, see Selection Result 0x01 – User Selection Request aborted, cancelled by user 0x02 – User Selection Request aborted, timeout
selection	Indicates the menu item selected by the user. This is a single byte zero based binary value.

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

4.57 cancelTransaction

This function cancels a transaction while waiting for the user to insert a card.

```
(int) cancelTransaction
```

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

The library returns the status of this function in the delegate method **onEMVCommandResult (EMV Device Only)**.

Parameters:

Parameter	Description
status	Result codes: 0x0000 = Success, the transaction was cancelled 0x038D = Failure, no transaction currently in progress 0x038F = Failure, transaction in progress, card already inserted

4.58 setAcquirerResponse (EMV Device Only)

This function informs EMV device to process transaction decision from acquirer.

```
(int) setAcquirerResponse:(Byte*) response length:(int) length
```

Parameters:

Parameter	Description
response	Hex string for the response data following TLV response message. For details about the ARQC format, see the <i>Programmer's Manual (COMMANDS)</i> for the specific device with which you are communicating.
length	Two byte binary, most significant byte first. This gives the total length of the Acquirer Response message that follows.

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

4.59 sendExtendedCommand (EMV Device and iDynamo 5 Gen III Only)

Send extended command to device. Command response is returned in the event message `onDeviceExtendedResponseReceived`.

```
(int) sendExtendedCommand:(NSString *)Command
```

Parameters:

Parameter	Description
Command	Hexadecimal string of the byte array for the extended command. The first two bytes represent the value of the extended command. The next two bytes (most significant byte first) indicate the total length of the following data in bytes.

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

4.60 sendCommandSync

Send a synchronous command to device. Operation waits for the command to return.

```
(NSString) sendCommandSync:(NSString *)Command
```

Parameters:

Parameter	Description
Command	Hexadecimal string of the byte array for the command.

Return Value: String containing the response data of the command.

4.61 sendExtendedCommandSync (EMV Device Only)

Send a synchronous extended command to device. Operation waits for the command to return.

```
(NSString) sendExtendedCommandSync:(NSString *)Command
```

Parameters:

Parameter	Description
Command	Hexadecimal string of the byte array for the extended command. The first two bytes represent the value of the extended command. The next two bytes (most significant byte first) indicate the total length of the following data in bytes.

Return Value: String containing the response data of the command.

4.62 setTimeout

Set the timeout in milliseconds for a sync command or sync extended command. Default value is 5000 milliseconds.

```
(void) setTimeout:(NSUInteger)timeoutMS
```

Parameters:

Parameter	Description
timeoutMS	Timeout in milliseconds.

Return Value: None.

4.63 setTimeFrame

Set the time frame (delay) for run loop or background thread. Default value is 10 milliseconds.

```
(void) setTimeFrame:(unsigned int)ms.
```

Parameters:

Parameter	Description
ms	Time frame delay in milliseconds.

Return Value: None.

4.64 enableDebugPrint

Enable the debug print from SDK. Default value is NO.

```
(void) enableDebugPrint:(BOOL)enabled.
```

Parameters:

Parameter	Description
enabled	When set to true, the SDK prints debugging messages in the XCode debug console. Debug messages consist of data sent to and received from the device. This may include iAP2.

Return Value: None.

4.65 updateFirmware

Updates the device firmware

```
(int) updateFirmware: (int) firmwareType
Data : (NSData*) firmwareData;
```

Parameters:

Parameter	Description
firmwareType	Type of firmware to update. <ul style="list-style-type: none"> 1 - Main Firmware
firmwareData	Firmware data.

Return Value:

- 0 = UPDATE_FIRMWARE_STARTED
- 1 = UPDATE_FIRMWARE_SUCCESS
- 9 = UPDATE_FIRMWARE_ERROR
- 15 = UPDATE_FIRMWARE_NA

4.66 sendNFCCCommand

This function sends a command to an NFC tag type 2. The NFC tag must first be activated by calling startTransaction() with NFC enabled.

```
(int) sendNFCCCommand: (NSString*) command
lastCommand : (BOOL) lastCommand
encrypt : (BOOL) encrypt;
```

Parameters:

Parameter	Description
command	<p>Command to send to the NFC tag.</p> <ul style="list-style-type: none"> • Get Version • Read • Fast Read • Write • Compatibility Write • Read_Cnt • PWD_Auth • Read_Sig
lastCommand	<p>Determines if this is the last NFC command to complete the operation.</p> <p>true = This is the last command. Device will provide a single beep after receiving a successful response from the NFC tag. To send subsequent commands, the NFC tag must be activated by calling startTransaction() with NFC enabled.</p> <p>false = Expect more commands (Default).</p> <p>Either set to true or false, if the NFC tag command fails, device will provide a double beep.</p>
encrypt	<p>Determines if data returned is to be encrypted.</p> <p>true = Encrypt data</p> <p>false = Do not encrypt data (Default)</p>

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

4.67 sendClassicNFCCommand

This function sends a command to a NFC Mifare Classic Tag type 2. The NFC tag must first be activated by calling startTransaction() with NFC enabled.

```
(int) sendClassicNFCCommand: (NSString*) command
lastCommand : (BOOL) lastCommand
encrypt : (BOOL) encrypt;
```

Parameters:

Parameter	Description
command	Command to send to the NFC tag.

Parameter	Description
lastCommand	<p>Determines if this is the last NFC command to complete the operation.</p> <p><code>true</code> = This is the last command. Device will provide a single beep after receiving a successful response from the NFC tag. To send subsequent commands, the NFC tag must be activated by calling <code>startTransaction()</code> with NFC enabled.</p> <p><code>false</code> = Expect more commands (Default).</p> <p>Either set to <code>true</code> or <code>false</code>, if the NFC tag command fails, device will provide a double beep.</p>
encrypt	<p>Determines if data returned is to be encrypted.</p> <p><code>true</code> = Encrypt data</p> <p><code>false</code> = Do not encrypt data (Default)</p>

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

4.68 sendDESFireNFCCCommand

This function sends a command to an NFC Mifare DESFire Light Tag Type 4. The NFC tag must first be activated by calling `startTransaction()` with NFC enabled.

```
(int) sendDESFireNFCCCommand: (NSString*) command
lastCommand : (BOOL) lastCommand
encrypt : (BOOL) encrypt;
```

Parameters:

Parameter	Description
command	<p>Command to send to the NFC tag. See DESFire Data Sheet (MF2DLHX0). Should follow ISO 7816-4 APDU format.</p>
lastCommand	<p>Determines if this is the last NFC command to complete the operation.</p> <p><code>true</code> = This is the last command. Device will provide a single beep after receiving a successful response from the NFC tag. To send subsequent commands, the NFC tag must be activated by calling <code>startTransaction()</code> with NFC enabled.</p> <p><code>false</code> = Expect more commands (Default).</p> <p>Either set to <code>true</code> or <code>false</code>, if the NFC tag command fails, device will provide a double beep.</p>

Parameter	Description
encrypt	Determines if data returned is to be encrypted. true = Encrypt data false = Do not encrypt data (Default)

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

4.69 sendNFCCommandSync

This function sends a command to an NFC tag type 2. The NFC tag must first be activated by calling startTransaction() with NFC enabled.

```
(NSString*) sendNFCCommandSync: (NSString*) command
lastCommand : (BOOL) lastCommand
encrypt : (BOOL) encrypt;
```

Parameters:

Parameter	Description
command	Command to send to the NFC tag. <ul style="list-style-type: none"> • Get Version • Read • Fast Read • Write • Compatibility Write • Read_Cnt • PWD_Auth • Read_Sig
lastCommand	Determines if this is the last NFC command to complete the operation. true = This is the last command. Device will provide a single beep after receiving a successful response from the NFC tag. To send subsequent commands, the NFC tag must be activated by calling startTransaction() with NFC enabled. false = Expect more commands (Default). Either set to true or false, if the NFC tag command fails, device will provide a double beep.
encrypt	Determines if data returned is to be encrypted. true = Encrypt data false = Do not encrypt data (Default)

Return Value:

- Extended response of the NFC command.

4.70 sendNFCCCommandAsync

This function sends a command to an NFC tag type 2. The NFC tag must first be activated by calling startTransaction() with NFC enabled.

```
(int) sendNFCCCommandAsync: (NSString*) command
lastCommand : (BOOL) lastCommand
encrypt : (BOOL) encrypt
response : (void (^)(NSString*)) response;
```

Parameters:

Parameter	Description
command	<p>Command to send to the NFC tag.</p> <ul style="list-style-type: none"> • Get Version • Read • Fast Read • Write • Compatibility Write • Read_Cnt • PWD_Auth • Read_Sig
lastCommand	<p>Determines if this is the last NFC command to complete the operation.</p> <p>true = This is the last command. Device will provide a single beep after receiving a successful response from the NFC tag. To send subsequent commands, the NFC tag must be activated by calling startTransaction() with NFC enabled.</p> <p>false = Expect more commands (Default).</p> <p>Either set to true or false, if the NFC tag command fails, device will provide a double beep.</p>
encrypt	<p>Determines if data returned is to be encrypted.</p> <p>true = Encrypt data</p> <p>false = Do not encrypt data (Default)</p>
response	<p>Callback to receive the response.</p>

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

5 MTSCRA Delegate Methods

After issuing the methods in section 3 **Important Information About Bluetooth LE**, the MTSCRA SDK libraries will call these Delegate methods (callback functions) to provide the requested data and / or a detailed response. For details about data received by these functions, see the *Programmer's Manual (COMMANDS)* for the specific device with which you are communicating.

For details about registering Delegate methods, see the demo application included with the SDK.

5.1 trackDataReadyNotification

The SDK sends this notification when card data is available from the device.

5.2 devConnectionNotification

The SDK sends this notification when the connection status of the device changes.

5.3 onDataReceived

Return a card object type with card swipe data.

Parameter	Description
cardDataObj	MTCardData object
instance	Instance ID

5.4 cardSwipeDidStart

Card swipe has started.

Parameter	Description
instance	Instance ID

5.5 cardSwipeDidGetTransError

Card swipe got an error during transmission.

5.6 onDeviceConnectionDidChange

Device connection changed whether from close to open or vice versa.

Parameter	Description
deviceType	MTSCRADeviceType object
connected	Boolean for connection state. True = Connected False = Not connected
instance	Instance ID

5.7 bleReaderConnected

Bluetooth LE Reader was connected.

Parameter	Description
peripheral	CBPeripheral object

5.8 bleReaderDidDisconnected

Bluetooth LE Reader was disconnected.

Parameter	Description
connectionInfo	MTConnectionInfo object

5.9 bleReaderDidDiscoverPeripheral

Bluetooth LE Reader was discovered.

5.10 bleReaderStateUpdated

Bluetooth LE state changed. Enumerated possible values that would arrive via this delegate are:

Parameter	Description
state	MTSCRABLEState

5.11 onDeviceResponse

This message occurs when a non EMV command response is returned from the device.

Parameter	Description
data	Byte array containing the data received from the device. For details, see the <i>Programmer's Manual (COMMANDS)</i> for the specific device with which you are communicating.

5.12 onDeviceError

This message occurs when an error occurs.

Parameter	Description
error	Error object.

5.13 onTransactionStatus (EMV Device Only)

The SDK sends this notification when the transaction status has changed.

Parameter	Description
data	Byte array containing the transaction status received from the device. For details about the Transaction Status, see the <i>Programmer's Manual (COMMANDS)</i> for the specific device with which you are communicating.

5.14 onDisplayMessageRequest (EMV Device Only)

Device request for displaying information to user.

Parameter	Description
data	Byte array containing the display message received from the device. For details about the Display Message Request, see the <i>Programmer's Manual (COMMANDS)</i> for the specific device with which you are communicating.

5.15 onUserSelectionRequest (EMV Device Only)

Device request for application to display a User Selection Menu.

Parameter	Description
data	Byte array containing the display selection from the device. For details about the Display Message Request, see the <i>Programmer's Manual (COMMANDS)</i> for the specific device with which you are communicating.

5.16 onARQCReceived (EMV Device Only)

This notification is sent from the device for ARQC data.

Offset	Field Name	Value
0	Message Length	Two byte binary, most significant byte first. This gives the total length of the ARQC message that follows.
2	ARQC Message	For details about the ARQC format, see the <i>Programmer's Manual (COMMANDS)</i> for the specific device with which you are communicating.

5.17 onTransactionResult (EMV Device Only)

This message occurs when the transaction result is received from the EMV device.

Parameter	Description
data	Byte array containing the transaction result from the EMV device. For details about the Display Message Request, see the <i>Programmer's Manual (COMMANDS)</i> for the specific device you are communicating with.

5.18 onEMVCommandResult (EMV Device Only)

This message occurs when an EMV command result is received from the EMV device.

Parameter	Description
data	Byte array containing the command result from the EMV device. For details about the Display Message Request, see the <i>Programmer's Manual (COMMANDS)</i> for the specific device you are communicating with.

5.19 onDeviceExtendedResponseReceived

This message occurs when and extended response is received from the device.

Parameter	Description
data	Hexadecimal string containing the extended response data received from the device. The first two bytes represent the result codes for the extended command. The next two bytes (most significant byte first) indicate the total length for the following data in bytes.

5.20 deviceNotPaired

This message occurs when a command is sent to an unpaired Bluetooth LE device.

5.21 didGetRSSI

This message occurs when and extended response is received from the device.

Parameter	Description
RSSI	Integer value for the Received Signal Strength Indicator.

Parameter	Description
error	NSError object

5.22 debugInfoCallback

This call back will be raised if host software subscribes to it.

6 MTSCRA Interfaces

6.1 MTCARDData

```
@interface MTCARDData : NSObject

- (id)initWithCardData:(NSString*)cardData;
/*!
  @attribute cardIIN
  @discussion cardIIN is 6 digits of the account, usually can help to identify the issuer
  */
@property(nonatomic, strong) NSString *cardIIN;
/*!
  @attribute cardData
  @discussion track1 | track2 | track3
  */
@property(nonatomic, strong) NSString *cardData;
/*!
  @attribute cardLast4
  @discussion last 4 digits of account, usually can help user to identify him/her self
  */
@property(nonatomic, strong) NSString *cardLast4;
/*!
  @attribute cardName
  @discussion card holder name
  */
@property(nonatomic, strong) NSString *cardName;

@property (strong, nonatomic) NSString *cardLastName;
@property (strong, nonatomic) NSString *cardMiddleName;
@property (strong, nonatomic) NSString *cardFirstName;

@property(nonatomic, strong) NSString *cardExpDate;
@property(nonatomic, strong) NSString *cardServiceCode;
@property(nonatomic, strong) NSString *cardStatus;
@property(nonatomic, strong) NSString *responseData;
@property(nonatomic, strong) NSString *maskedTracks;
@property(nonatomic, strong) NSString *encryptedTrack1;
@property(nonatomic, strong) NSString *encryptedTrack2;
@property(nonatomic, strong) NSString *encryptedTrack3;
@property(nonatomic, strong) NSString *encryptionStatus;
@property(nonatomic, strong) NSString *maskedTrack1;
@property(nonatomic, strong) NSString *maskedTrack2;
@property(nonatomic, strong) NSString *maskedTrack3;
@property(nonatomic, strong) NSString *trackDecodeStatus;
@property(nonatomic, strong) NSString *encryptedMagneprint;
@property(nonatomic, strong) NSString *magneprintStatus;
@property(nonatomic, strong) NSString *deviceSerialNumber;
@property(nonatomic, strong) NSString *deviceSerialNumberMagTek;
```

DynaMAX, eDynamo, aDynamo, uDynamo, iDynamo 5 Gen II, iDynamo 5 Gen III, iDynamo 6, kDynamo, sDynamo, tDynamo | Secure Card Reader Authenticators | SDK Programmer's Manual (iOS)

```
@property(nonatomic, strong) NSString *encryptedSessionID;
/*!
  @attribute deviceKSN
  @discussion Key Serial Number for the card swipe
 */
@property(nonatomic, strong) NSString *deviceKSN;
@property(nonatomic, strong) NSString *deviceFirmware;
@property(nonatomic, strong) NSString *deviceName;
@property(nonatomic, strong) NSString *deviceCaps;
@property(nonatomic, strong) NSString *deviceStatus;
@property(nonatomic, strong) NSString *tlvVersion;
@property(nonatomic, strong) NSString *devicePartNumber;
@property(nonatomic, strong) NSString *capMSR;
@property(nonatomic, strong) NSString *capTracks;
@property(nonatomic, strong) NSString *capMagStripeEncryption;
@property(nonatomic, strong) NSString *maskedPAN;
@property(nonatomic) long cardPANLength;
@property(nonatomic, strong) NSString *additionalInfoTrack1;
@property(nonatomic, strong) NSString *additionalInfoTrack2;
@property(nonatomic, strong) NSString *responseType;
@property(nonatomic) long batteryLevel;
@property(nonatomic) long swipeCount;
@property(nonatomic, strong) NSString *firmware;
@property(nonatomic, strong) NSString *tagValue;
@property(nonatomic) int magnePrintLength;
@property(nonatomic) int cardType;
@property(nonatomic, strong) NSString *cardExpDateMonth;
@property(nonatomic, strong) NSString *cardExpDateYear;
@property(nonatomic, strong) NSString *cardPAN;
@property(nonatomic, strong) NSString *track1DecodeStatus;
@property(nonatomic, strong) NSString *track2DecodeStatus;
@property(nonatomic, strong) NSString *track3DecodeStatus;
@property(nonatomic, strong) NSDate* timeStamp;

@property(nonatomic, strong) NSString* messageID;
@property(nonatomic, strong) NSString* msrDUKPTKeyInfo; // 16 bytes
@property(nonatomic, strong) NSString* mpDUKPTKeySerialNumber; // 0/20/24
@property(nonatomic, strong) NSString* mpDUKPTKeyInfo; // 16 bytes
@property(nonatomic, strong) NSString* macKeyInfo; // 16 bytes
@property(nonatomic, strong) NSString* macMessageLength;
@property(nonatomic, strong) NSString* mac;

@property(nonatomic, strong) NSString* tokenDUKPTKeyInfo;
@property(nonatomic, strong) NSString* tokenDUKPTKeySerialNumber;
@property(nonatomic, strong) NSString* encryptedQwantumDataBuffer;
@property(nonatomic, strong) NSString* qwantumStatus;
@property(nonatomic, strong) NSString* qwantumToken;
@property(nonatomic, strong) NSString* qwantumCardID;

@property(nonatomic, strong) NSString* customerMessageCode;
```

```
@property(nonatomic, strong) NSString* customerMessage;

@property(nonatomic) BOOL isQwantumCard;
@property(nonatomic) BOOL isQwantumBuffer;
@property(nonatomic) BOOL isCustomerMessage;

@property(nonatomic, strong) NSString* encryptedSCDE;
@property(nonatomic, strong) NSString* scdeDUKPTKeySerialNumber;
@property(nonatomic, strong) NSString* scdeDUKPTKeyInfo;

@property(nonatomic, strong) NSDictionary* allObjects;
@property(nonatomic, strong) NSArray<NSString*>* objectArray;

@end
```


A.2 Set MSR Head Always On 0x58

The host uses this command to directly control power to the magnetic stripe reader head inside the device, to manage device power consumption and battery life. The host must power up the head before a cardholder swipes a magnetic stripe card outside the scope of an EMV transaction.

When the state is set to **Always On**, the head will always be powered.

When the state is set to **Off When Idle**, the head will be off when the device is idle. In this state, the device will not be able to read magnetic stripe cards outside the scope of EMV transactions without first turning the head on.

If the host sends startTransaction() the device will automatically power up the head, then power it down when the EMV transaction terminates.

Offset	Field Name	Value
0	State	One byte specifying what the state should be: 0x00 = Off When Idle 0x01 = Always On

Result codes: 0x00 = Success

Example Request (Hex):

```
NSString *SetMSRALwaysOn = @"580101";
[self.mtSCRALib sendCommandSync:SetMSRALwaysOn];
```

```
NSString *SetMSROffWhenIdle = @"580100";
[self.mtSCRALib sendCommandSync:SetMSROffWhenIdle];
```

Appendix B Code Examples

B.1 Open Device

```
self.mtSCRALib = [[MTSCRA alloc] init];
[self.mtSCRALib
listenForEvents:(TRANS_EVENT_OK|TRANS_EVENT_START|TRANS_EVENT_ERROR)];

//iDynamo
[self.mtSCRALib setDeviceType:(MAGTEKIDYNAMO)];
[self.mtSCRALib setDeviceProtocolString:@"com.magtek.idynamo"];

//Audio
//[self.mtSCRALib setDeviceType:(MAGTEKAUDIOREADER)];
[self.mtSCRALib openDevice];
```

B.2 Close Device

```
[self.mtSCRALib closeDevice];
```

B.3 Get Track Data from Reader

```
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(trackDataReady:) name:@"trackDataReadyNotification"
object:nil];

- (void)trackDataReady:(NSNotification *)notification
{
    NSNumber *status = [[notification userInfo]
valueForKey:@"status"];
    [self performSelectorOnMainThread:@selector(onDataEvent:)
withObject:status waitUntilDone:YES];
}

- (void)onDataEvent:(id)status
{
    //[self clearLabels];
    switch ([status intValue]) {

        case TRANS_STATUS_OK:
            NSLog(@"TRANS_STATUS_OK");
            break;

        case TRANS_STATUS_ERROR:
            NSLog(@"TRANS_STATUS_ERROR");
            break;

        default:
            break;
    }
}
```

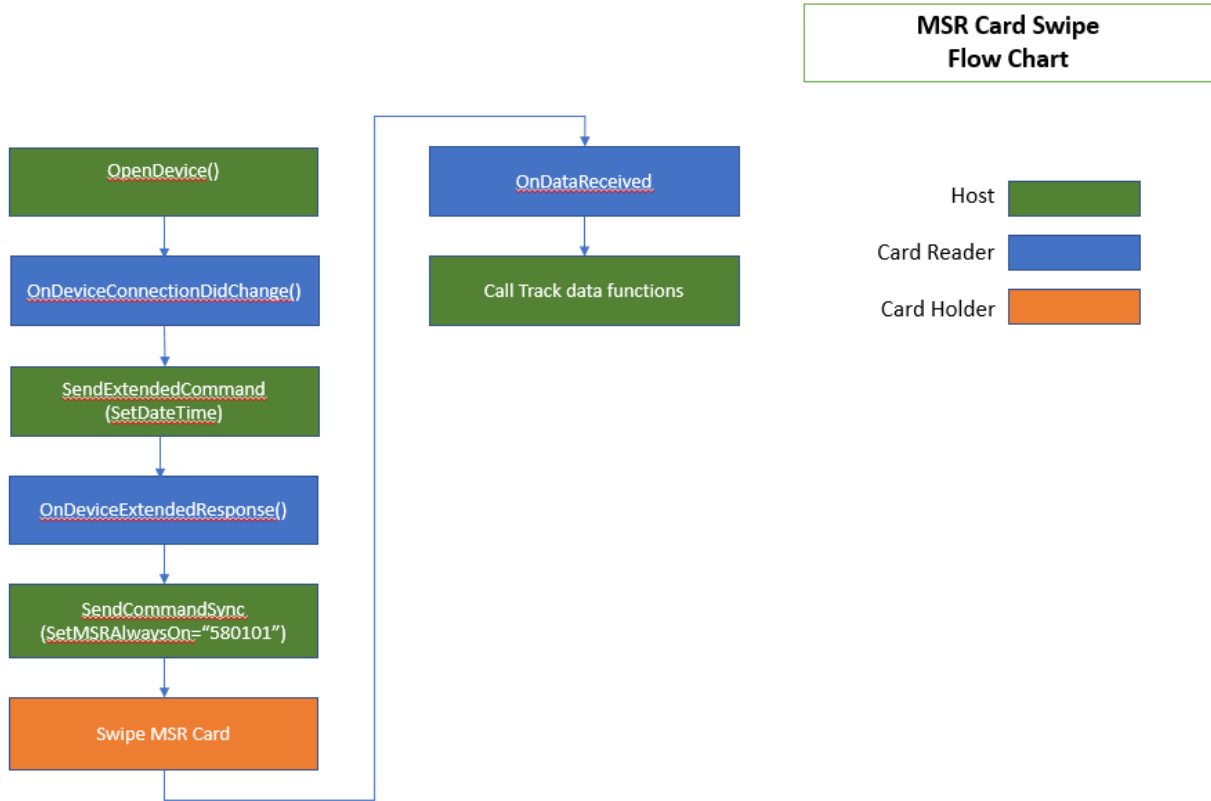
B.4 Get Connection Status of Reader

```
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(devConnStatusChange)
name:@"devConnectionNotification" object:nil];

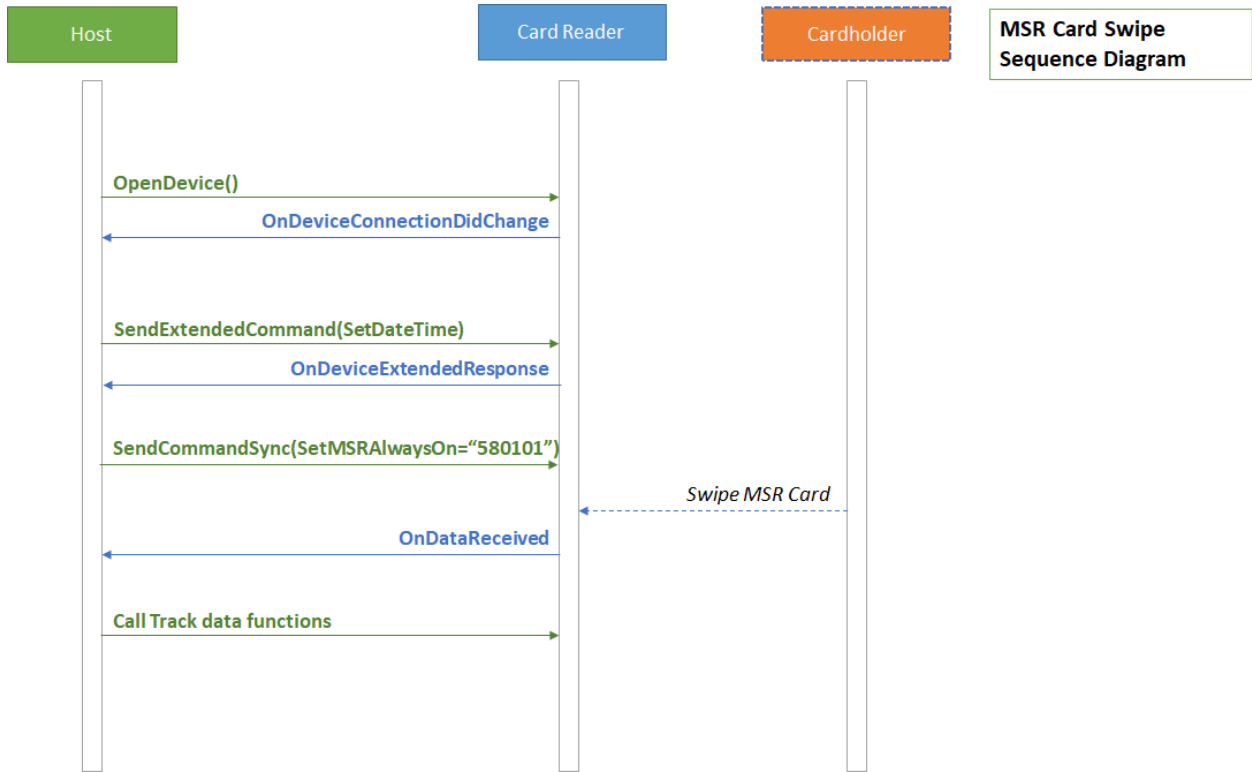
- (void)devConnStatusChange
{
    BOOL isDeviceConnected = [self.mtSCRALib isDeviceConnected];
    if (isDeviceConnected)
    {
        self.deviceStatus.text = @"Device Connected";
    }
    else
    {
        self.deviceStatus.text = @"Device Disconnected";
    }
}
```

B.5 Diagrams

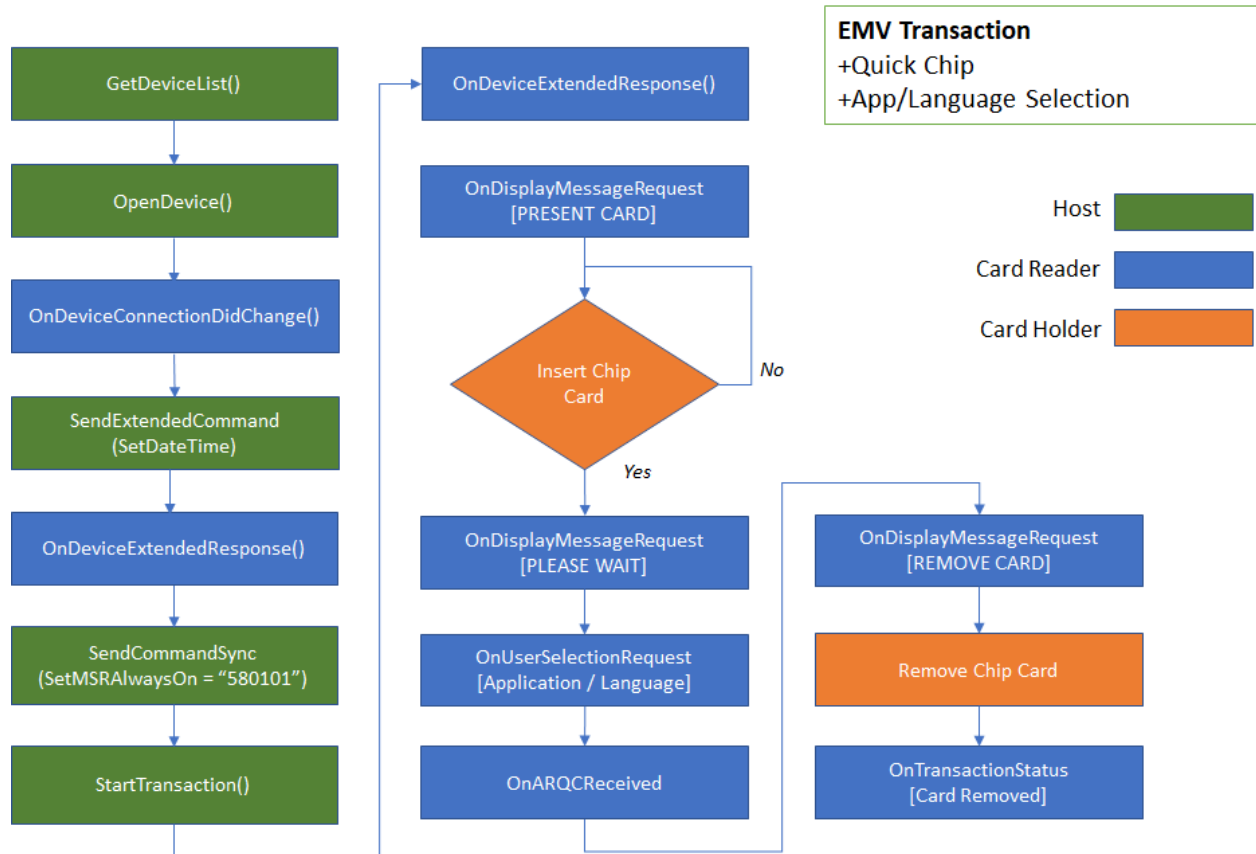
B.5.1 MSR Card Swipe



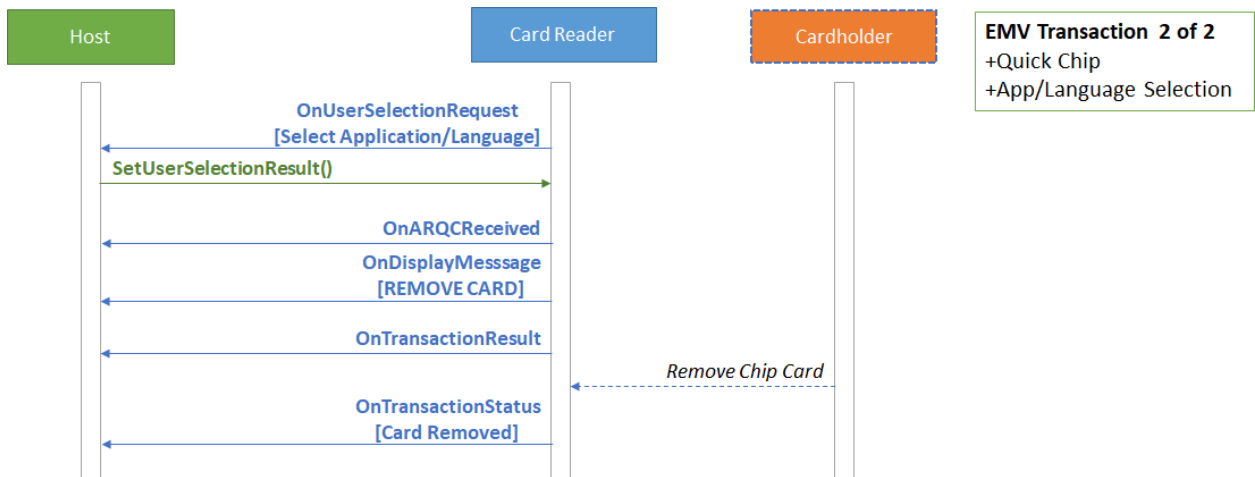
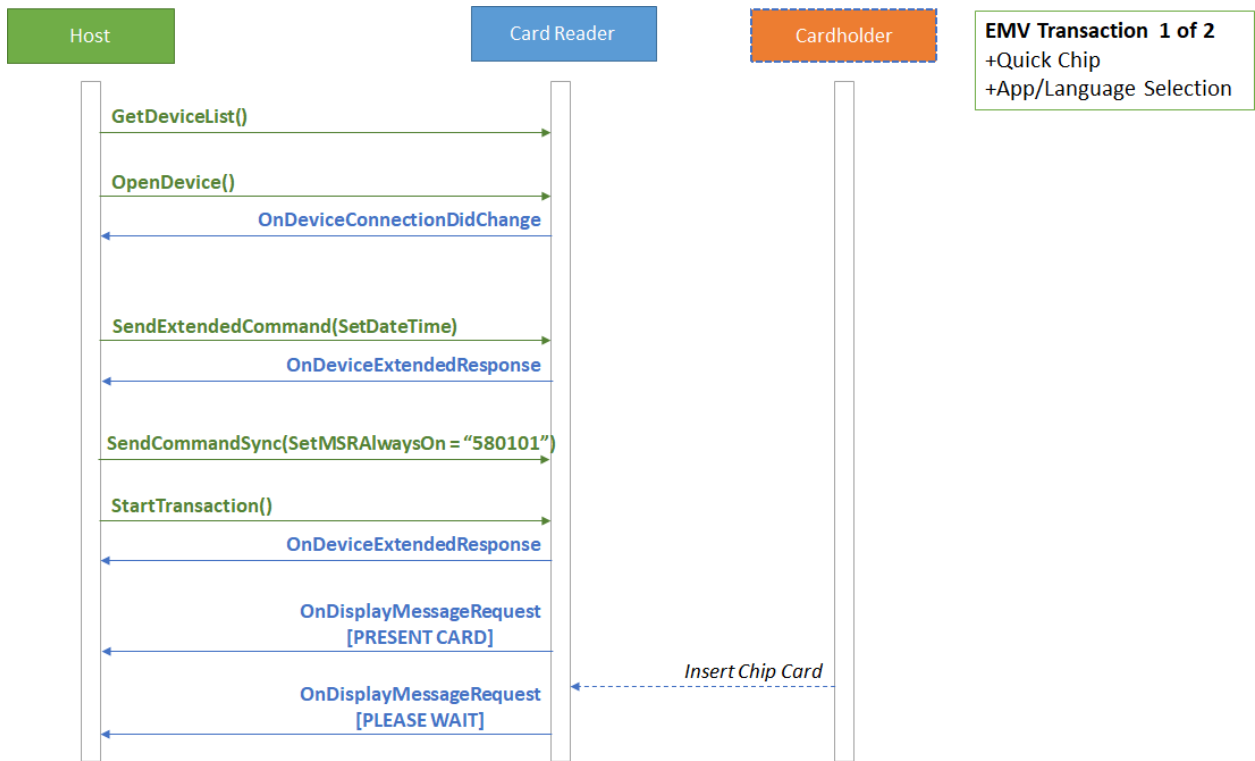
Note: This is broad to cover multiple devices. For devices that do not need or support a specific command, (i.e. `SetMSRAAlwaysOn`), that portion may be omitted in the flow.



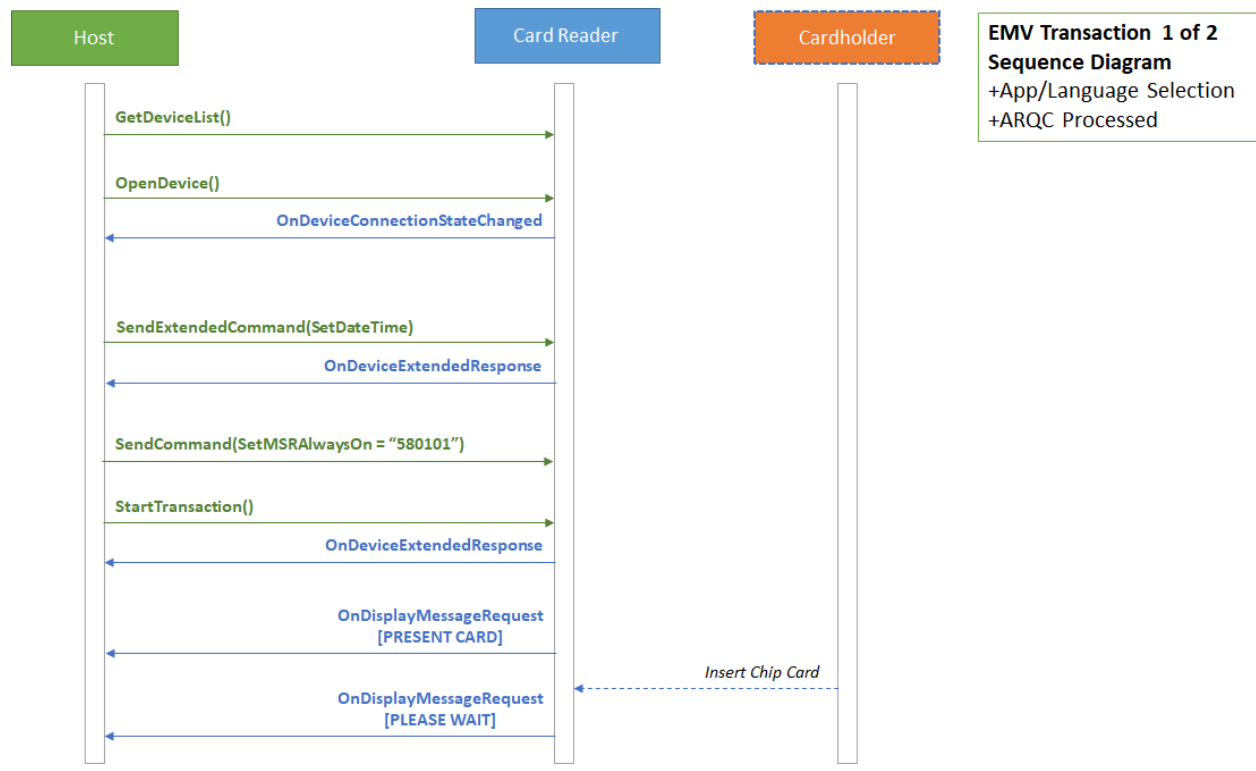
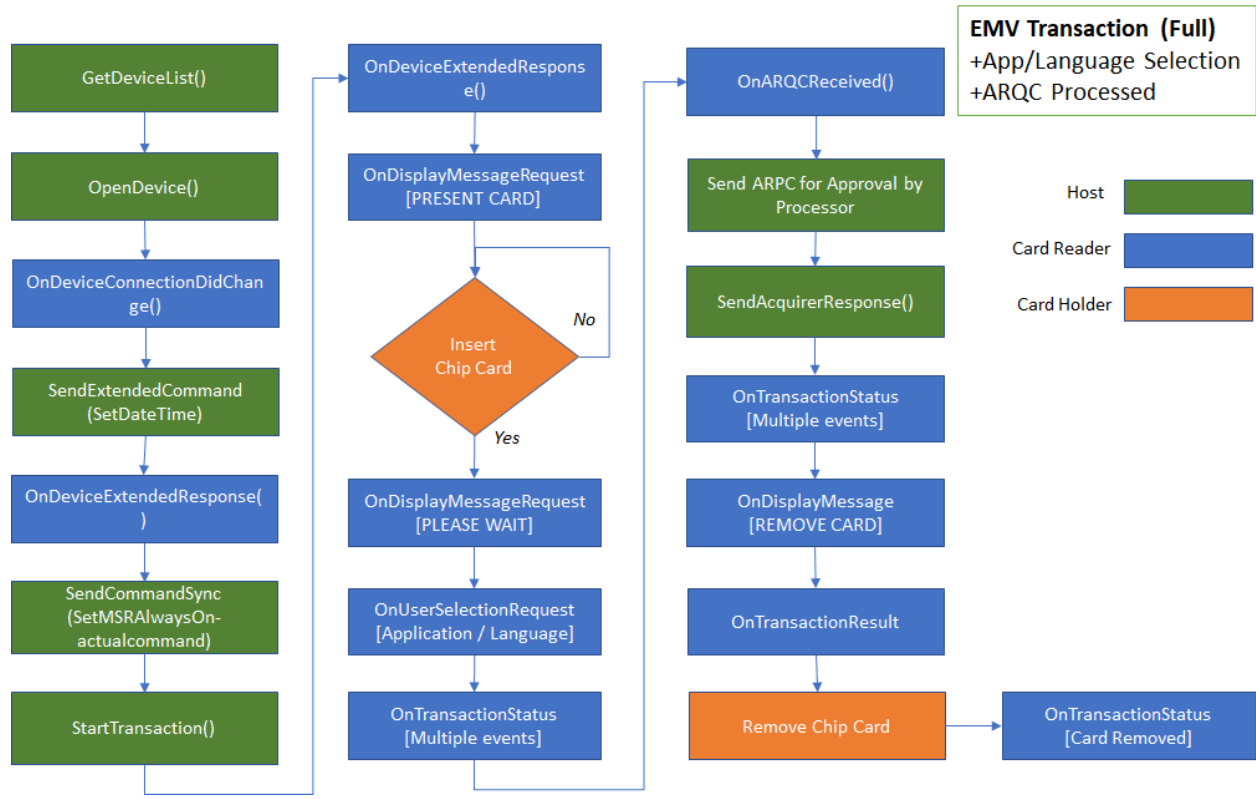
B.5.2 EMV Transaction (Quick Chip)

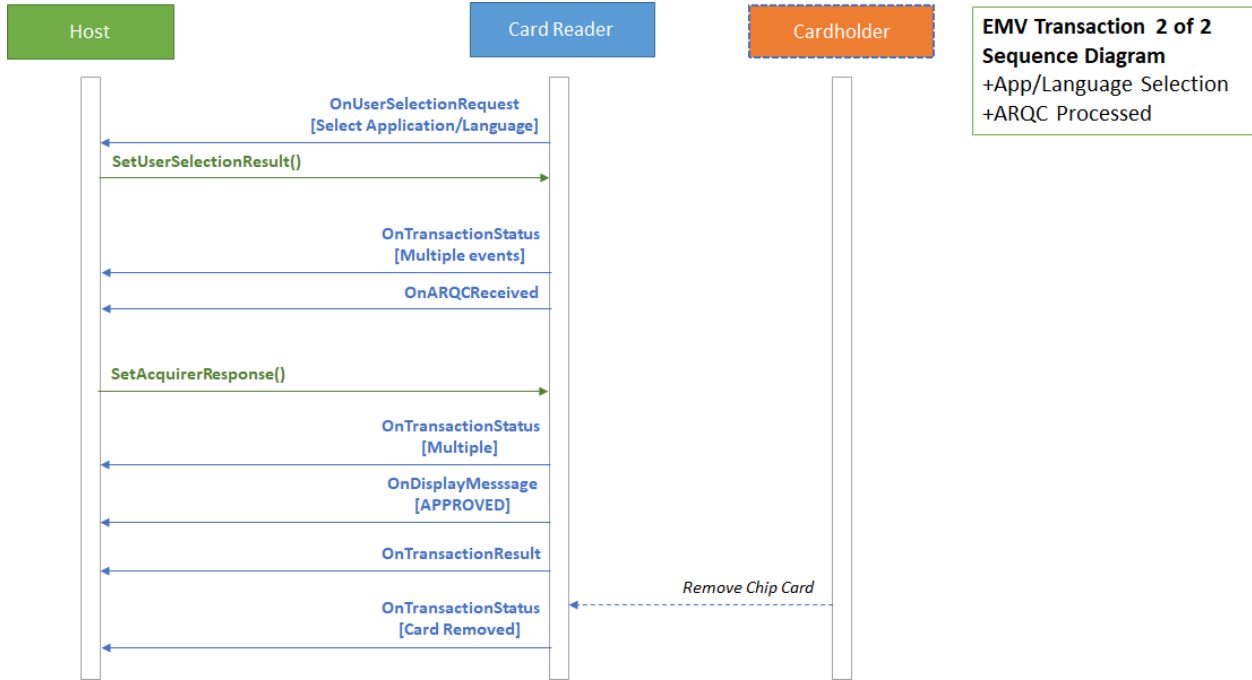


Note: This is broad to cover multiple devices. For devices that do not need or support a specific command, (i.e. SetMSRAAlwaysOn), that portion may be omitted in the flow.



B.5.3 EMV Transaction (Full)





Note: This is broad to cover multiple devices. For devices that do not need or support a specific command, (i.e. SetMSRAAlwaysOn), that portion may be omitted in the flow.

Appendix C Supported Devices

Feature / Product	cDynamo	DynaMAX	eDynamo	iDynamo 5	iDynamo 5 (Gen II)	iDynamoe 5 (Gen III)	iDynamo 6	kDynamo	sDynamo	tDynamo	uDynamo
MSR Swipe	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MSR Insert	N	N	N	N	N	N	N	N	N	N	N
MSR 3 Tracks	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
MSR Disable	Y	N	N	Y	N	N	N	N	N	N	N
MSR Swap Tracks 1/3	N	N	N	N	N	N	N	N	N	N	N
MSR Embedded V5 Head	N	N	N	N	Y	Y	Y	Y	Y	Y	N
MSR Configurable MSR Variants	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
MSR Configurable MP Variants	N	Y	Y	N	N	Y	Y	Y	N	Y	N
MSR SureSwipe	N	Y	Y	N	N	N	N	N	N	N	N
MSR JIS Capable	Y	N	N	Y	N	N	N	N	Y	N	N
SHA-1	N	Y	Y	N	N	Y	N	N	N	N	N
SHA-256	N	N	N	N	N	Y	N	N	N	N	N
Configurable SHA	N	Y	Y	N	N	N	N	N	N	N	N
Configurable Encryption Algorithm	N	N	N	N	N	Y	Y	N	N	N	N
Set Mask Service Code	N	N	N	N	N	N	N	N	N	N	N
Never Mask Service Code	N	N	Y	Y	Y	Y	Y	Y	Y	Y	N
MagneSafe 2.0	N	N	Y	N	N	N	N	N	N	N	N
EMV Contact	N	N	Y	N	N	N	Y	Y	N	Y	N
EMV Contactless	N	N	N	N	N	N	Y	Y	N	Y	N
EMV Offline ODA	N	N	Y	N	N	N	N	N	N	N	N
EMV MSR Flow	N	N	N	N	N	N	Y	Y	N	Y	N
EMV Contact Quick Chip	N	N	Y	N	N	N	Y	Y	N	Y	N
EMV Contactless Quick Chip	N	N	N	N	N	N	Y	Y	N	Y	N
External PIN Accessory Support	N	N	N	N	N	N	Y	N	N	N	N
Keypad Entry	N	N	N	N	N	N	N	N	N	N	N
Fixed Key	N	N	N	N	N	Y	N	N	N	N	N
Secondary DUKPT Key	N	Y	Y	N	N	Y	N	N	N	N	Y
Power Mgt Scheme (PM#)	N	2	3	N	N	N	7	5	N	5	4
Battery-Backed RTC	N	N	Y	N	N	Y	N	N	N	N	N
OEM Features	N	N	N	N	N	N	N	N	N	N	N

Feature / Product	cDynamo	DynaMAX	eDynamo	iDynamo 5	iDynamo 5 (Gen II)	iDynamoe 5 (Gen III)	iDynamo 6	kDynamo	sDynamo	tDynamo	uDynamo
Transaction Validation	N	N	N	N	N	N	N	N	N	N	N
Display	N	N	N	N	N	N	N	N	N	N	N
Multi-Language	N	N	Y	N	N	N	Y	Y	N	Y	N
Tamper	N	N	Y	N	N	Y	N	N	N	N	N
Extended Commands	N	N	Y	N	N	Y	Y	Y	N	Y	N
Extended Notifications	N	N	Y	N	N	Y	Y	Y	N	Y	N
Dual USB Ports	N	N	N	N	N	Y	Y	N	N	Y	N
Pairing Modes	N	N	Y	N	N	N	N	N	N	Y	N
Custom Advertising	N	N	Y	N	N	N	N	N	N	Y	N
Configurable Lightning FID	Y	N	N	N	Y	Y	Y	Y	N	N	N
Auxiliary Ports	N	N	N	N	N	N	N	N	N	N	N
External LED Control	N	N	N	N	N	Y	N	N	N	N	N
Encrypt Bulk Data (b)	120	24	24	120	N	Y	N	N	N	N	24

Appendix D Enums

D.1 Error Codes

ERROR_SUCCESS = 0
ERROR_TIMEOUT = 1
ERROR_DEVICE_NOT_OPEN = 5
ERROR_INVALID_PARAMETER = 6
ERROR_DEVICE_COMMUNICATION_ERROR = 7
ERROR_OTHER_ERROR = 9
ERROR_BUSY = 15
ERROR_DATA_IS_NOT_EXIST = 16
ERROR_NOT_SUPPORT = 17
ERROR_UNKNOWN = 255

D.2 MTSCRADeviceType

MAGTEKAUDIOREADER = Audio readers aDynamo, uDynamo.
MAGTEKIDYNAMO = iOS 30-pin and Lightning readers iDynamo.
MAGTEKDYNAMAX = Bluetooth LE reader DynaMAX.
MAGTEKEDYNAMO = Bluetooth LE reader eDynamo
MAGTEKUSBMSR = USB on OSX Only
MAGTEKKDYNAMO = Lightning EMV reader kDynamo
MAGTEKTDYNAMO = Bluetooth LE reader tDynamo
MAGTEKDYNAWAVE = DynaWave
MAGTEKMDYNAMO = mDynamo
MAGTEKIDYNAMO_G3 = USB-C iDynamo 5 Gen III
MAGTEKNONE

D.3 MTSCRATransactionStatus

TRANS_STATUS_OK = Transaction succeeded.
TRANS_STATUS_START = Reader started sending data.
TRANS_STATUS_ERROR = Reader failed sending data.

D.4 MTSCRATransactionEvent

TRANS_EVENT_OK = Transaction succeeded.
TRANS_EVENT_ERROR = Reader failed sending data.
TRANS_EVENT_START = Reader started sending data.

D.5 MTSCRATransactionData

TLV_OPSTS = Operation Status
TLV_CARDSTS = Card Information
TLV_TRACKSTS = Card tracks status
TLV_CARDNAME = Cardholder name
TLV_CARDIIN = Card issuer identification number
TLV_CARDLAST4 = Last four digits of PAN number
TLV_CARDEXPDATE = Card Expiration date
TLV_CARDSVCCODE = Card service code
TLV_CARDPANLEN = Length of the PAN
TLV_ENCTK1 = Encrypted track 1
TLV_ENCTK2 = Encrypted track 2

TLV_ENCTK3 = Encrypted track 3
TLV_DEVSN = Device serial number
TLV_DEVSNMAGTEK = Device serial number created by MagTek
TLV_DEVFW = Device firmware version
TLV_DEVNAME = Device model name
TLV_DEVCAPS = Device capabilities
TLV_DEVSTATUS = Device status
TLV_TLVVERSION = Firmware TLV version
TLV_DEVPARTNUMBER = Device part number
TLV_CAPMSR = Magstripe capabilities
TLV_CAPTRACKS = Track capabilities
TLV_CAPMAGSTRIPEENCRYPTION = Magstripe encryption capabilities
TLV_KSN = KSN
TLV_CMAC = CMAC
TLV_SWPCOUNT = Swipe count
TLV_BATTLEVEL = Battery level
TLV_CFGTLVVERSION = TLV version
TLV_CFGDISCOVERY = Discovery
TLV_CFGCARDNAME = Card name
TLV_CFGCARDIIN = Card issuer identification number
TLV_CFGCARDLAST4 = Card last 4 PAN
TLV_CFGCARDEXPDATE = Card expiration date
TLV_CFGCARDSVCCODE = Card service code
TLV_CFGCARDPANLEN = Card PAN length
TLV_MSCTK1 = Masked Track 1
TLV_MSCTK2 = Masked Track 2
TLV_MSCTK3 = Masked Track 3
TLV_HASHCODE = Hash code
TLV_SESSIONID = Session ID
TLV_MAGNEPRINT = MagnePrint
TLV_MAGNEPRINT_STS = MagnePrint status

D.6 MTSCRACapabilities

CAP_MASKING = Masking
CAP_ENCRYPTION = Encryption
CAP_CARD_AUTH = Card authorization
CAP_DEVICE_AUTH = Device authorization
CAP_SESSION_ID = Session ID
CAP_DISCOVERY = Discovery

D.7 ConnectionTypes

BLE = Bluetooth LE
BLE_EMV = Bluetooth LE EMV
USB = Universal Serial Bus
Lightning = Lightning
NONE = None

D.8 DebugDomain

Connection
CommandMessageData

CardMessageData
BLERawMessage
DeviceInfo
RawDataMessage

D.9 MTSCRABLEState

OK
OFF
RESETTING
DISCONNECTED
UNSUPPORTED
UNAUTHORIZED
UNKNOWN