# User Manual

# Elyctis Multi Slot Reader – USB

240 rue de Gernelle
84120 PERTUIS
FRANCE
Phone +33 6 27 71 37 68
Fax +33 4 92 82 27 42
E-mail: contact@elyctis.fr

**1 CARD READER SUITE – TEST APPLICATION ..... 3**

1.1 FIRMWARE UPGRADE ..... 3
1.2 PC/SC DIAGNOSTICS ..... 3
1.3 BINARY CONFIGURATION ..... 3

**2 PC/SC 2.0 ..... 5**

2.1 HOW TO ACCESS CONTACT/CONTACTLESS CARDS VIA PC/SC? ..... 5
*2.1.1 Establish Context ..... 5*
*2.1.2 List Readers ..... 5*
*2.1.3 Connect ..... 5*
*2.1.4 Data and Command transfer with Card ..... 6*
*2.1.5 Disconnect ..... 6*
*2.1.6 Release ..... 6*
2.2 ATR GENERATION FOR CONTACTLESS CARDS ..... 7
*2.2.1 CPU Cards ..... 7*
*2.2.2 Storage Cards ..... 7*

**3 ACCESSING READER THROUGH PC/SC ..... 8**

3.1 READER CONTROL COMMANDS ..... 8
*3.1.1 Get Firmware Version ..... 9*
*3.1.2 Get Serial Number ..... 9*

**4 SMART CARD TEST COMMANDS ..... 10**

4.1 COMMAND TRANSPORTATION ..... 10
4.2 TEST COMMANDS FOR CONTACTLESS INTERFACE ..... 12
*4.2.1 ISO-14443-2 test command ..... 12*
*4.2.2 Transmit pattern and receive 14443 command ..... 12*
*4.2.3 ISO-14443-3 test command ..... 13*
*4.2.4 ISO-14443-4 test command ..... 15*
4.3 TEST COMMANDS FOR CONTACT INTERFACE ..... 17
*4.3.1 Select slot ..... 17*
*4.3.2 Issue Power-On ..... 17*
*4.3.3 Issue PPS ..... 18*
*4.3.4 Set ETU ..... 18*
*4.3.5 Set BWT ..... 18*
*4.3.6 Set CWT ..... 18*
*4.3.7 Set GT ..... 19*
*4.3.8 Send Data ..... 19*
*4.3.9 Issue Power-Off ..... 19*
*4.3.10 Warm Reset ..... 20*

# 1 Card Reader Suite – Test Application

CardRdrSuite is a test application that is provided along with Identive's Multi Slot USB Reader/Writer for customer use. CardRdrSuite consists of several sub-applications covering the following functions

## *1.1 Firmware Upgrade*

Reader firmware can be downloaded into the reader/writer using this sub-application. Salient features are

- Auto reader detection
- User friendly GUI
- Provision to view device's current firmware version and the version of the firmware to be downloaded

The application operates in two modes

- DFU class mode
  - Disaster recovery mode to recover reader if the reader is plugged out accidentally during firmware download
  - Individual DFU driver to handle this mode

- Proprietary mode
  - Firmware upgrade support without using the DFU driver

Refer to section "Firmware Upgrade" of Card Reader Suite help file for detailed description. Help file can be launched using the "Help" button in the main window of the Card Reader Suite application.

## *1.2 PC/SC Diagnostics*

Any Card APDU or Pseudo APDU can be issued and tested using this sub-application.

Salient features are

- APDU execution from script file (.APDU)
- Easy to edit script files
- Error status logging in a file
- Working sample scripts for each card type supported

Refer to section "PC/SC Diagnostics" of the Card Reader Suite help file for detailed description

## *1.3 Binary configuration*

All configurable parameters of the reader can be modified using these sub-applications. These applications will configure the firmware binary file only.

**Configure Binary**

- Configures
  - USB Chapter 09 parameters
  - Firmware Version parameters
  - Hardware Version parameters
  - Other firmware configurations
- Provides option to make the parameters non-modifiable after configuring

Refer to section "Configure binary" of Card Reader Suite help file for detailed description

**Edit Binary**

- Configures
    - RF parameters
    - Mifare keys stored in the non-volatile memory of the reader
    - DESFire keys stored in the non-volatile memory of the reader
- Provides option to make the parameters non-modifiable after configuring

Refer to section “Edit Binary” of Card Reader Suite help file for detailed description

# 2 PC/SC 2.0

"Identive's Multi Slot USB Reader/Writer" can be accessed using the standard PC/SC architecture for communicating with cards. This makes card access easy, as it uses the same communication procedure for all the cards.

The Microsoft Developer Network (MSDN) library gives valuable information and a detailed description of all SCard APIs for communicating with the reader, through the Windows PC/SC framework (WINSCARD APIs – Refer [R4]).

## *2.1 How to Access Contact/Contactless Cards via PC/SC?*

Contact/Contactless cards can be easily accessed through PC/SC using standard WINSCARD smart card API's for card access. The following steps provide guidelines for using the PC/SC compliant WINSCARD API's to access cards using the Identive's Multi Slot USB Reader/Writer.

### 2.1.1 Establish Context

Calling the SCardEstablishContext function is the first step. This API will initialize all other PC/SC APIs and allocate all resources necessary for a smart card session. Also this function establishes the resource manager context (the scope) within which database operations is performed.

```
LONG SCardEstablishContext ( IN DWORD dwScope,
                             IN LPCVOID pvReserved1,
                             IN LPCVOID pvReserved2,
                             OUT LPSCARDCONTEXT phContext);
```

### 2.1.2 List Readers

The next step is to get a list of all PC/SC readers connected to the system using the SCardListReaders function. Look for "**ELYCTIS CL reader** <*16 byte serial number*> <*x*>" for Contactless Slot and "**ELYCTIS CNT reader** <*16 byte serial number*> <*x*> <*y*> <*z*>" for Contact slot in the list returned. If multiple devices are connected to your system, they will be enumerated with different serial numbers. Where x, y and z are numbers appended to the name by the driver. This is in order to make the PC/SC name unique for all devices.

**Example:**
"ELYCTIS CL reader 0000000000000001 0",
"ELYCTIS CNT reader 0000000000000001 0 0 0",
"ELYCTIS CNT reader 0000000000000001 0 1 0",
"ELYCTIS CNT reader 0000000000000001 0 2 0",
"ELYCTIS CNT reader 0000000000000001 0 3 0",

could be the list returned when the following function is executed.

```
LONG SCardListReaders ( IN SCARDCONTEXT *phContext,
                        IN LPCTSTR mszGroups,
                        OUT LPTSTR mszReaders,
                        IN OUT LPDWORD pcchReaders);
```

**Note:** The OUT parameter phContext of SCardEstablishContext is used as in parameter for this API.

### 2.1.3 Connect

Next step is to connect to the card via the reader/writer. The SCardConnect API establishes a connection (using a specific resource manager context) between the calling application and a smart card contained by the specific reader. If no card exists in the specified reader, an error is returned.

```
LONG SCardConnect (  IN SCARDCONTEXT *phContext,
                     IN LPCTSTR szReader,
                     IN DWORD dwShareMode,
                     IN DWORD dwPreferredProtocols,
                     OUT LPSCARDHANDLE phCard,
                     OUT LPDWORD pdwActiveProtocol);
```

**Note:** The OUT parameter phContext of SCardEstablishContext is used as in parameter for this API.

### 2.1.4 Data and Command transfer with Card

Command and data that are transferred to the card are called as APDUs (application protocol data unit) in smart card terminology. The SCardTransmit function sends a service request to the smart card, and expects data back from the card.

```
LONG SCardTransmit ( IN SCARDHANDLE *phCard,
                     IN LPC/SCARD_I0_REQUEST pioSendPci,
                     IN LPCBYTE pbSendBuffer,
                     IN DWORD cbSendLength,
                     IN OUT LPSCARD_IO_REQUEST pioRecvPci,
                     OUT LPBYTE pbRecvBuffer,
                     IN OUT LPDWORD pcbRecvLength);
```

**Note:** The OUT parameter phCard of SCardConnect is used as in parameter for this API.

### 2.1.5 Disconnect

It is not mandatory to disconnect the card after the completion of all transactions, but it is recommended. The SCardDisconnect function terminates a connection previously opened between the calling application and a smart card in the target reader.

```
LONG SCardDisconnect ( IN SCARDHANDLE *phCard,
                       IN DWORD dwDisposition);
```

**Note:** The OUT parameter phCard of SCardConnect is used as in parameter for this API.

### 2.1.6 Release

This step ensures that all system resources are being released. The SCardReleaseContext function closes an established resource manager context, freeing any resources allocated under that context.

```
LONG SCardReleaseContext (IN SCARDCONTEXT *phContext);
```

**Note:** The OUT parameter phContext of SCardEstablishContext is used as in parameter for this API.

## *2.2 ATR Generation for Contactless cards*

To make contactless cards available within the PC/SC framework, the Identive's Multi Slot USB Reader/Writer generates a PC/SC compliant ATR according to PC/SC v2.01.09 "Interoperability Specification for ICCs and Personal Computer Systems" (reference [R2])

### 2.2.1 CPU Cards

The ATR of Contactless processor cards are composed as described in PC/SC v2.01.09, Part3: Requirements for PC connected Interface Devices, 3.1.3.2.3.1, Table 3.5

### 2.2.2 Storage Cards

The ATR of storage cards (i.e. cards without CPU) are composed as described in PC/SC v2.01.09, Part3: Requirements for PC connected Interface Devices, 3.1.3.2.3.2, Table 3.6. In order to allow the HOST application to identify a storage card type properly, its standard and card name is mapped according to the Part3: Supplement Document of PC/SC v2.01.04

**Note:** The registered Application Provider Identifier (RID) returned by the Identive's Multi Slot USB Reader/Writer for storage cards (cards without CPU) is A0 00 00 03 06, which is the RID of the PC/SC workgroup

# 3 Accessing Reader through PC/SC

In some specific situations, PC/SC specifications are not enough to cover the whole functional field. This happens typically,

- When working with memory cards or even microprocessor based cards not following the ISO-7816-4 standard (APDU formalism)
- When needing to perform actions on to the reader/writer itself, and not onto the card (Like modifying the RF parameters of the reader/writer)

In order to cover the above two cases, proprietary ISO-7816 wrappers are supported. These are called Pseudo APDUs. Refer to [R7] in order to understand the basic structure of APDU

Pseudo-APDUs covered in this section pertains to

- Modification/Retrieval of RF Parameters into/from the reader's memory; **Knowledge on ISO-14443 specification is a mandate.**
- Modification/Retrieval of Register contents pertaining to the RFID reader silicon; **Knowledge on RFID Reader Silicon is a mandate.**

And Pseudo-APDUs pertaining to modification/access of reader PIN and card keys are covered in section 6.

**Note:**

- All command and response bytes are sent and received as hexadecimal values
- Reader specific pseudo APDU commands can be sent via the PC/SC interface by either with the presence/absence of card. When the card is in the reader field, the pseudo APDU can be sent to the device with the normal SCardConnect API in SHARED mode, whereas if the card is not in the field, it can be sent to the device using a SCardConnect API in DIRECT mode. A sample code for using the DIRECT mode is given in Appendix A
- Both of these modes are possible in WinXP OS. Whereas **in WinCE 5.0, DIRECT mode is not supported**. **An alternative solution in WinCE OS or in platforms where DIRECT mode is not supported is that these commands can be sent via the GUID interface. For more information refer Appendix B.**
- In the pseudo APDUs described in this section, specifying a value of 00 in the **Le** field indicates maximum number of available response bytes from the card, as described in reference [R7]

## *3.1 Reader Control Commands*

The Reader/Writer control commands are used to modify the reader/writer parameters in order to fine tune the reader/writer's performance or to suit the end applications requirements. It follows standard ISO-7816-4 (as per reference [R7]) command/ response format.

The Reader/Writer control commands have the following general APDU format

**C-APDU format:**

| *CLA* | *INS* | *P1* | *P2* | *Lc* | *Data* | | *Le* |
|---|---|---|---|---|---|---|---|
| FF | 00 | 00 | 00 | Number of bytes in Data field | Reader/Writer Control Command | Command Parameters | 00 |

**R-APDU format:**

| *Data* | **Status Word** | |
|---|---|---|
| Reader/Writer Control Response | SW1 | SW2 |

For possible values and description of status word, refer Table 8.

The following are the Reader/Writer control commands:

### 3.1.1 Get Firmware Version

Get Firmware Version command is used to retrieve the firmware version of the reader/writer.

**C-APDU:**

| *CLA* | *INS* | *P1* | *P2* | *Lc* | *Data* | | | *Le* |
|---|---|---|---|---|---|---|---|---|
| FF | 00 | 00 | 00 | 03 | 01 | 08 | 00 | 00 |

**R-APDU:**

| *Data* | **Status** |
|---|---|
| Firmware Version Record | 9000 |

**Firmware version record structure:**

| *Byte 1* | *Byte 2* | *Byte 3* | *Byte 4* | *Byte 5* | *Byte 6* | *Byte 7* | *Byte 8* |
|---|---|---|---|---|---|---|---|
| 08 | 00 | xx | xx | Firmware Version (MN) | Firmware Version (MJ) | xx | xx |

From the above firmware version record structure, one can get the firmware version as **MJ.MN** (i.e., **MJ** stands for Major version; **MN** stands for Minor version)

**Example:**
**Command:** FF 00 00 00 03 01 08 00 00
**Response:** 08 00 01 00 11 01 FF FF 90 00
(i.e., the firmware version is 01.11)

**Note:**

- The command fails if the reader/writer configuration is invalid
- The command fails if any of the command byte is invalid

### 3.1.2 Get Serial Number

Get Serial Number command is used to retrieve the USB Serial number of the reader/writer. The USB Serial Number is of 32 bytes and it takes the ASCII character format.

**C-APDU:**

| *CLA* | *INS* | *P1* | *P2* | *Lc* | *Data* | | | *Le* |
|---|---|---|---|---|---|---|---|---|
| FF | 00 | 00 | 00 | 03 | 01 | 3C | 00 | 00 |

**R-APDU:**

| *Data* | **Status** |
|---|---|
| USB Serial Number Record | 9000 |

**USB Serial number record structure:**

| *Byte 1* | *Byte 2* | *Byte 3* | *Byte 4* | *Byte 5 – Byte 36* |
|---|---|---|---|---|
| 24 | 00 | xx | xx | USB Serial Number (32 byte ASCII characters in Hex format) |

**Example:**
**Command:** FF 00 00 00 03 01 3C 00 00
**Response:** 24 00 02 FF 30 31 30 31 30 31 30 31 30 34 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 31 90 00
(i.e., the USB serial number is “01010101040000000000000000000001”)

**Note:**

- The command fails if the reader/writer configuration is invalid
- The command fails if any of the command byte is invalid

# 4 Smart card test commands

Smart card test commands are used for testing the contactless and contact interface from the test application. Here the user's test application possesses full control over the reader and the reader's firmware will just pass through the commands sent by the application to the card.

The structure of the test commands follows the byte sequential command structure for smart cards, according to ISO-7816 APDUs. The APDUs are distinguished by their direction, to or from the PCD.

For each C-APDUs sent to the PCD by the test application, an R-APDU will be returned as a confirmation. R-APDUs will not be confirmed by the PCD.

C-APDUs are byte sequences consisting of two parts: Header and Body. The header only (first four bytes) is mandatory. The body is optional and may contain data with prior length indicator and/or length indicator for expected R-APDU. The card presence depends on the command and application case or context.

The response to a C-APDU will be returned in general as R-APDU. Optionally the R-APDU may contain data, but surely contains the status bytes SW1 & SW2.

The trailer of the R-APDU transports the result of an operation. It has to be interpreted byte wise. SW1 classifies the result in general and SW2 gives an exact value for the indicated error class. The following table depicts the supported values of SW1 & SW2.

| ***Class (SW1)*** | ***Description*** |
|---|---|
| '90' | Normal processing |
| '62', '63', '6C' | Warning |
| '64', '65' | Execution Error |
| '67' to '6F' | Checking Error |

***Table 7*** *– Return code classes*

The value of SW2 is class specific.

| ***Category*** | ***SW1*** | ***SW2*** | ***Description*** |
|---|---|---|---|
| **Success** | '90' | '00' | Command successful |
| **Warning** | '6C' | 'XX' | Le and available data are not same; 'XX' is the number of available data. If Le= 0x00, all available data is returned. |
| **Error** | '64' | '00' | Timeout, expecting response<br>no response within the time |
| | | '01' | Internal Error |
| | '67' | '00' | Wrong Length |
| | '68' | '00' | Class byte is not correct |
| | '6A' | '81' | INS not supported |
| | | '82' | Function is not supported |
| | '6B' | '00' | Wrong parameter P1-P2 |

***Table 9*** *– Common return codes*

## *4.1 Command Transportation*

Before sending the test commands to the device, it is recommended to stop the polling sequence of the device, by sending a CONTROL_POLLING command. For more information on sending this command via the PCSC and GUID interfaces, refer Appendix A and Appendix B respectively. An exception case is, if the user needs to send a Part-4 command to a contactless card (say DESFire); this step is not a mandate. i.e., he can very well issue the DESFire Part-4 commands without stopping the polling sequence, using the protocol description given under section 8.2.4.

There are two types of test commands namely,

- Contactless test commands
- Contact test commands

The reader can distinguish between the contactless test commands and the contact test commands using the CLA byte of the C-APDU. For instance, we use 0xFF as the CLA byte (clubbed with a set of INS byte values viz., 0x92 – 0x98) for the contactless test commands and 0x92 for the contact test commands.

**Note:** All RFU bits and bytes described here must be set to 0

## *4.2 Test commands for contactless interface*

The following are the test commands for the contactless interface. **In order to use these commands, knowledge on ISO/IEC 14443 specification is a mandate.**

- ISO-14443-2 test command
- Transmit pattern and receive 14443 command
- ISO-14443-3 test command
- ISO-14443-4 test command

### 4.2.1 ISO-14443-2 test command

This command is used to test the RF interface, the modulation index, framing and coding of the data, transmitted by the PCD. Testing with this command does not mandate the presence of a card but it may be inserted to the field in order to check the mutual induction to the magnetic field

**C – APDU:**

| *CLA* | *INS* | *P1* | *P2* | *Lc* | *Data in* | *Le* |
|---|---|---|---|---|---|---|
| 0xFF | 0x92 | xx | RFU | xx | xx | - |

**R – APDU:**

| Data Out | SW1SW2 |
|---|---|
| - | xxxx |

**P1 – Subcarrier and Data Coding Parameter**

| *b7* | *b6* | *b5* | *b4* | *b3* | *b2* | *b1* | *b0* | *Description* |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | ------ | | | | | | No carrier, RF is turned off |
| 0 | 1 | ------ | | | | | | No sub-carrier, just carrier, RF is turned on |
| 1 | 0 | ------ | | | | | | Carrier modulated with sub-carrier; if there are some bytes to transmit, Lc = n, n ≠ 0 |
| 1 | 1 | ------ | | | | | | RFU |
| ------ | | RFU | 0 | ------ | | | | ISO-14443A type transmission* |
| ------ | | | 1 | ------ | | | | ISO-14443B type transmission* |
| ------ | | | | RFU | | 0 | 0 | Transmission at 106kbps |
| ------ | | | | | | 0 | 1 | Transmission at 212kbps |
| ------ | | | | | | 1 | 0 | Transmission at 424kbps |
| ------ | | | | | | 1 | 1 | Transmission at 848kbps |

**Note:** * The transmission according to normal frame, which includes all framing e.g., start bit, stop bit, parity bit, SOF, EOF, CRC etc.

The following table lists the return codes in addition to the common return codes:

| *SW1SW2* | *Meaning* |
|---|---|
| '6A83' | Transmission type not supported |
| '6A84' | Transmission speed not supported |

### 4.2.2 Transmit pattern and receive 14443 command

This command transmits a bit pattern independent from any SCIC command structure and coding. The pattern can be used to measure modulation index, rise and fall times, overshoots etc. This command can be used to test by using a reference card to receive any pattern by the SCIC.

**C – APDU:**

| *CLA* | *INS* | *P1* | *P2* | *Lc* | *Data in* | *Le* |
|---|---|---|---|---|---|---|
| 0xFF | 0x94 | xx | xx | xx | Pattern | xx |

**R – APDU:**

| Data Out | SW1SW2 |
|---|---|
| - | xxxx |

**P1 – Transmit pattern and receive 14443**

| *b7* | *b6* | *b5* | *b4* | *b3* | *b2* | *b1* | *b0* | *Description* |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | - | - | - | - | RFU |
| - | - | - | - | - | - | - | 0 | ISO-14443A type Transceive data coding |
| - | - | - | - | - | - | - | 1 | ISO-14443B type Transceive data coding |
| | | | | xxx | | | - | Number of bits of last byte will be transmitted, 0 means all bits will be transmitted |

**Note:** The data in the 'data in' field is not interrupted; the complete data is sent to the air. No framing, e.g. start bit, stop bit, CRC, SOF, EOF is not added.

**P2 – Transmit Pattern and Receive (bit rate) 14443**

| *b7* | *b6* | *b5* | *b4* | *b3* | *b2* | *b1* | *b0* | *Description* |
|---|---|---|---|---|---|---|---|---|
| RFU | | - | - | - | RFU | 0 | 0 | Transmit @106 kbps |
| | | - | - | - | | 0 | 1 | Transmit @212 kbps |
| | | - | - | - | | 1 | 0 | Transmit @424 kbps |
| | | - | - | - | | 1 | 1 | Transmit @848 kbps |
| | | RFU | 0 | 0 | - | - | - | Receive @106 kbps |
| | | | 0 | 1 | | | | Receive @212 kbps |
| | | | 1 | 0 | | | | Receive @424 kbps |
| | | | 1 | 1 | | | | Receive @848 kbps |

**Return codes of Transmit Pattern and Receive 14443**

| *SW1SW2* | *Meaning* |
|---|---|
| '6A83' | Transmission type not supported |
| '6A84' | Transmission speed not supported |
| '6A87' | Different bit rate is not supported |
| '6A8A' | Modulation index is not supported |

## 4.2.3 ISO-14443-3 test command

This command transmits a 14443-3 command and returns the data received from the SCIC.

**C – APDU:**

| *CLA* | *INS* | *P1* | *P2* | *Lc* | *Data in* | *Le* |
|---|---|---|---|---|---|---|
| 0xFF | 0x96 | xx | xx | - | - | xx |

**R – APDU:**

| *Data Out* | *SW1SW2* |
|---|---|
| Response of the card | xxxx |

**P1 – Command Byte**

| *b7* | *b6* | *b5* | *b4* | *b3* | *b2* | *b1* | *b0* | *Description* |
|---|---|---|---|---|---|---|---|---|
| ------ | | | 0 | ISO-14443A type command | | | | |
| | | | | 0 | 0 | 0 | 1 | REQA |
| | | | | 0 | 0 | 1 | 0 | WUPA |
| | | | | 0 | 0 | 1 | 1 | HLTA |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | 0 | 1 | 0 | 0 | PCD does complete part 3 A type, returns UID+SAK |
| | | | 1 | Anti-collision is handled by user | | | |
| | | | | 0 | 0 | 1 | ANTICOLLISION Sel_level 1 |
| | | | | 0 | 1 | 0 | ANTICOLLISION Sel_level 2 |
| | | | | 0 | 1 | 1 | ANTICOLLISION Sel_level 3 |
| | | | | 1 | 0 | 0 | SELECT (Data in field must be 70+ last 4-byte UID + BCC) |
| | | | Other values are RFU | | | | |
| | | 1 | ISO-14443B type command | | | | |
| | | | 0 | 0 | 0 | 1 | REQB (P2 sets the number of slot) |
| | | | 0 | 0 | 1 | 0 | WUPB (P2 sets the number of slot) |
| | | | 0 | 0 | 1 | 1 | HLTB |
| | | | 0 | 1 | 0 | 0 | Slot-MARKER (slot number in P2) |
| | | | 0 | 1 | 0 | 1 | ATTRIB (P2 sets the communication speed) |
| | xx | ------ | | | | | Number of repetitions of the command |
| x | ------ | ------ | | | | | RFU |

**P2 – Coding of Bit rate for ATTRIB command**

| *b7* | *B6* | *b5* | *b4* | *b3* | *b2* | *b1* | *b0* | *Description* |
|---|---|---|---|---|---|---|---|---|
| RFU | | - | - | - | RFU | 0 | 0 | PCD to SCIC 106 kbps |
| | | - | - | - | | 0 | 1 | PCD to SCIC 212 kbps |
| | | - | - | - | | 1 | 0 | PCD to SCIC 424 kbps |
| | | - | - | - | | 1 | 1 | PCD to SCIC 848 kbps |
| | | RFU | 0 | 0 | - | - | - | SCIC to PCD 106 kbps |
| | | | 0 | 1 | | | | SCIC to PCD 212 kbps |
| | | | 1 | 0 | | | | SCIC to PCD 424 kbps |
| | | | 1 | 1 | | | | SCIC to PCD 848 kbps |
| CID | | -- | | | | | | Logical card identifier* |

**Note:** * Only four values of CID (0, 1, 2 & 3) are supported. For more info on CID, refer [R8], [R9] & [R10]

**P2 – Coding of Slot number of REQB/WUPB command**

| *b7* | *B6* | *b5* | *b4* | *b3* | *b2* | *b1* | *b0* | *Description* |
|---|---|---|---|---|---|---|---|---|
| RFU | | | | | xxx | | | Number of slots, N = $2^{(b2b1b0)}$; (b2b1b0 = 0 means N = $2^0$ = 1 |

**Note:** As multiple slots in contactless interface is not supported, the recommended value for N is 1

**P2 – Coding of slot number of Slot-MARKER command**

| *b7* | *B6* | *b5* | *b4* | *b3* | *b2* | *b1* | *b0* | *Description* |
|---|---|---|---|---|---|---|---|---|
| RFU | | | | 0 | 0 | 0 | 1 | Slot Number = 2 |
| | | | | 0 | 0 | 1 | 0 | Slot Number = 3 |
| | | | | .. | .. | .. | .. | ……. |
| | | | | 1 | 1 | 1 | 1 | Slot Number = 16 |

**Note:** As multiple slots in contactless interface is not supported, it is recommended not to issue this command.

The return codes in addition to the common return codes are listed in the following table:

| *SW1SW2* | *Meaning* |
|---|---|
| '6A85' | The command is not supported |
| '6A86' | The repetition is not allowed |
| '6A87' | Different bit rate is not supported |
| '6A88' | Requested buffer size is bigger than the PCD buffer size |

## 4.2.4 ISO-14443-4 test command

This command transmits a 14443-4 command and returns the data received from the SCIC.

**C – APDU:**

| *CLA* | *INS* | *P1* | *P2* | *Lc* | *Data in* | *Le* |
|---|---|---|---|---|---|---|
| 0xFF | 0x98 | xx | xx | - | - | xx |

**R – APDU:**

| *Data Out* | *SW1SW2* |
|---|---|
| Response of the card** | xxxx |

** If there is a card response SW1SW2, this is considered as the 'Response of the card' data, e.g. if a 'select file' command returns '6A82', here the complete R-APDU will be '6A829000', similarly if the card returns '9000', the complete R-APDU will be '90009000'.

**P1 is structured as follows**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 | *Description* |
|---|---|---|---|---|---|---|---|---|
| RFU | | | | | | 0 | 0 | RATS (Type A only), P2 codes the parameter (FSDI and CID) according to ISO-14443-4 |
| | | | | | | 0 | 1 | PPS (Type A only), P2 codes the communication speed |
| | | | | | | 1 | 0 | The complete data of 'Data in' field is transmitted to the card, user must add PCB and CID according to T=CL protocol. |
| | | | | | | 1 | 1 | The complete data of 'Data in' field is transmitted to the card, PCD must take care of the PCB and CID, only I block is allowed in this mode. |

**Note:** P2 is coded according to the value in P1 or it would be simply '00'.

**Note:** CRC is calculated by the PCD and appended to the end of the frame.

**P2 – Communication speed for PPS command**

| *b7* | *b6* | *b5* | *b4* | *b3* | *b2* | *b1* | *b0* | *Description* |
|---|---|---|---|---|---|---|---|---|
| - | - | - | - | - | RFU | 0 | 0 | PCD to SCIC 106 kbps |
| - | - | - | - | - | | 0 | 1 | PCD to SCIC 212 kbps |
| - | - | - | - | - | | 1 | 0 | PCD to SCIC 424 kbps |
| - | - | - | - | - | | 1 | 1 | PCD to SCIC 848 kbps |
| - | - | RFU | 0 | 0 | -- | | | SCIC to PCD 106 kbps |
| - | - | | 0 | 1 | -- | | | SCIC to PCD 212 kbps |
| - | - | | 1 | 0 | -- | | | SCIC to PCD 424 kbps |
| - | - | | 1 | 1 | -- | | | SCIC to PCD 848 kbps |
| RFU | | -- | | | | | | --- |

**P2 – RATS command**

| *b7* | *b6* | *b5* | *b4* | *b3* | *b2* | *b1* | *b0* | *Description* |
|---|---|---|---|---|---|---|---|---|
| FSDI | | | | CID | | | | FSDI is according to the following table |

**Note:** * Only four values of CID (0, 1, 2 & 3) are supported. For more info on CID, refer [R11] and [R12]

**Coding of FSDI**

| **FSDI** | '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' – 'F' |
|---|---|---|---|---|---|---|---|---|---|---|
| **FSD** | 16 | 24 | 32 | 40 | 48 | 64 | 96 | 128 | 256 | RFU |

Return codes of ISO-14443-4 Test Command are as follows,

| ***SW1SW2*** | ***Meaning*** |
|---|---|
| '6A87' | Different bit rate is not supported |
| '6A88' | Requested buffer size is bigger than the PCD buffer size |

## *4.3 Test commands for contact interface*

The following are the test commands for the contact interface. **In order to use some of these commands, knowledge on ISO/IEC-7816 contact specification is a mandate.**

- Select slot
- Issue Power-On
- Issue PPS
- Set ETU
- Set BWT
- Set CWT
- Set GT
- Send Data
- Issue Power-Off
- Warm Reset

It is recommended to issue Le for all of the above commands. If the response length is unknown, the value can be marked as 0x00.

The return codes for all of the above commands are listed in **Common return codes** of section-7

### 4.3.1 Select slot

This command is used to select a contact slot using the logical slot number.

**C – APDU:**

| *CLA* | *INS* | *P1* | *P2* | *Lc* | *Data in* | *Le* |
|---|---|---|---|---|---|---|
| 0x91 | 0x00 | xx | xx | - | - | xx |

**Note:**

- As a general case, the first slot (SLOT_0) is the contactless interface; and the further slots (SLOT_1 to SLOT_xx) are the contact interfaces.
- The **Logical slot number 1** is equivalent to **Contact slot number 0**; **Logical slot number 2** is equivalent to **Contact slot number 1**; and so on.
- The logical slot number should be patched in the 'Slot' field of the CCID header in this command in order to set the correct slot for further communication.
- It is recommended to use the same logical slot number for rest of the commands also.

**R – APDU:**

| *Data Out* | *SW1SW2* |
|---|---|
| - | xxxx |

### 4.3.2 Issue Power-On

This command issues the Power-On sequence to the card and thereby gets the ATR of the card. The class switching will be happened in the device itself.

**C – APDU:**

| *CLA* | *INS* | *P1* | *P2* | *Lc* | *Data in* | *Le* |
|---|---|---|---|---|---|---|
| 0x91 | 0x01 | xx | xx | - | - | xx |

**R – APDU:**

| *Data Out* | *SW1SW2* |
|---|---|
| xx (ATR) | xxxx |

### 4.3.3 Issue PPS

Issues PPS command is used to switch the speed of the card to a different speed. It is an optional command.

**C – APDU:**

| *CLA* | *INS* | *P1* | *P2* | *Lc* | *Data in* | | | | *Le* |
|---|---|---|---|---|---|---|---|---|---|
| 0x91 | 0x02 | xx | xx | 0x04 | PPSS | PPS0 | PPS1 | PCK | xx |

The 4 bytes in the 'Data in' field corresponds to the command bytes of the PPS command. For more information, refer [R6]

**Note:** PPS2 & PPS3 are not supported

**R – APDU:**

| *Data Out* | *SW1SW2* |
|---|---|
| xx (PPS response) | xxxx |

### 4.3.4 Set ETU

Set ETU command is used to switch the speed of the card to a different speed.

**C – APDU:**

| *CLA* | *INS* | *P1* | *P2* | *Lc* | *Data in* | *Le* |
|---|---|---|---|---|---|---|
| 0x91 | 0x03 | xx | 00 | - | - | xx |

**R – APDU:**

| *Data Out* | *SW1SW2* |
|---|---|
| - | xxxx |

**P1 – Set ETU command**

| *b7* | *b6* | *b5* | *b4* | *b3* | *b2* | *b1* | *b0* | *Description* |
|---|---|---|---|---|---|---|---|---|
| Fi | | | | Di | | | | For more details on Fi & Di values, refer [R6] |

### 4.3.5 Set BWT

Set BWT command is used to set the block waiting time in the device. For calculating the value of BWT, refer [R6]. This is applicable only for T=1.

**C – APDU:**

| *CLA* | *INS* | *P1* | *P2* | *Lc* | *Data in* | *Le* |
|---|---|---|---|---|---|---|
| 0x91 | 0x04 | 00 | 00 | 04 | BWT (4 Bytes, LSB first) | xx |

**Note:** The calculated value should be in the units of etu's

**R – APDU:**

| *Data Out* | *SW1SW2* |
|---|---|
| - | xxxx |

### 4.3.6 Set CWT

Set CWT command is used to set the character waiting time in the device. For calculating the value of CWT, refer [R6].

**C – APDU:**

| *CLA* | *INS* | *P1* | *P2* | *Lc* | *Data in* | *Le* |
|---|---|---|---|---|---|---|
| 0x91 | 0x05 | 00 | 00 | 04 | CWT (4 Bytes, LSB first) | xx |

**Note:** The calculated value should be in the units of etu's

**R – APDU:**

| *Data Out* | *SW1SW2* |
|---|---|
| - | xxxx |

## 4.3.7 Set GT

Set GT command is used to set the guard time in the device. For calculating the value of GT, refer [R6].

**C – APDU:**

| *CLA* | *INS* | *P1* | *P2* | *Lc* | *Data in* | *Le* |
|---|---|---|---|---|---|---|
| 0x91 | 0x06 | <GT> | 00 | - | - | xx |

**Note:** The calculated value should be in the units of etu's

**R – APDU:**

| *Data Out* | *SW1SW2* |
|---|---|
| - | xxxx |

## 4.3.8 Send Data

Send data command is used to send the data to the card either with protocol T=0 or T=1. Here the exact data frame needs to be constructed by the test application and sent to the device. The device just pass-through the data bytes to the card and sends back the card's response to the host without any processing.

**C – APDU:**

| *CLA* | *INS* | *P1* | *P2* | *Lc* | *Data in* | *Le* |
|---|---|---|---|---|---|---|
| 0x91 | 0x07 | [0x00 for T=0] / [0x01 for T=1] | 0x00 | xx | xx | xx |

**R – APDU:**

| *Data Out* | *SW1SW2* |
|---|---|
| Response from the card | xxxx |

For T=0 protocol, the exact T0 frame (ISO-7816 frame format), which is intended by the card needs to be sent to the device in the 'Data in' field. For more information on T=0 protocol refer [R6]

Similarly for T=1 protocol, the exact T1 frame (including the T1 prologue and epilogue field), which is intended by the card needs to be sent to the device in the 'Data in' field. For more information on T=1 protocol refer [R6]

## 4.3.9 Issue Power-Off

This command issues the deactivating sequence to the card.

**C – APDU:**

| *CLA* | *INS* | *P1* | *P2* | *Lc* | *Data in* | *Le* |
|---|---|---|---|---|---|---|
| 0x91 | 0x08 | 00 | 00 | - | - | xx |

**R – APDU:**

| *Data Out* | *SW1SW2* |
|---|---|
| - | xxxx |

### 4.3.10 Warm Reset

This command issues the sequence described in the [R6, § 5.3.3]. The sequence starts with the ICC already powered.

**C – APDU:**

| *CLA* | *INS* | *P1* | *P2* | *Lc* | *Data in* | *Le* |
|---|---|---|---|---|---|---|
| 0x91 | 0x09 | 00 | 00 | - | - | xx |

**R – APDU:**

| *Data Out* | *SW1SW2* |
|---|---|
| - | xxxx |