Universal Serial Bus

Device Class Specification for
USB Chip/Smart Card Interface Devices

Revision 1.00
March 20, 2001
THIS SPECIFICATION IS PROVIDED “AS IS” WITH NO WARRANTIES WHATSOEVER INCLUDING ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION, OR SAMPLE.

A LICENSE IS HEREBY GRANTED TO REPRODUCE AND DISTRIBUTE THIS SPECIFICATION FOR INTERNAL USE ONLY. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY OTHER INTELLECTUAL PROPERTY RIGHTS IS GRANTED OR INTENDED HEREBY.

AUTHORS OF THIS SPECIFICATION DISCLAIM ALL LIABILITY, INCLUDING LIABILITY FOR NFRINGEMENT OF PROPRIETARY RIGHTS, RELATING TO IMPLEMENTATION OF INFORMATION IN THIS SPECIFICATION. AUTHORS OF THIS SPECIFICATION ALSO DO NOT WARRANT OR REPRESENT THAT SUCH IMPLEMENTATION(S) WILL NOT INFRINGE SUCH RIGHTS.

Contributors

Bob Nathan NCR
Stephen Chasko NCR
Don Chang Alcor Micro Inc
Daryl Adams Alcor Micro Inc
Michel Brun Philips Semiconductors
Christophe Chausset Philips Semiconductors
Jean Luc Luong Philips Semiconductors
Gerhard Hahn Cherry GmbH
Harald Folkenborn Cherry GmbH
Eric Vila ActivCard
Jerome Becquart ActivCard
Thierry Karlisch GemPlus
Francis Donnat GemPlus
Sergey Ivanov GemPlus
Jean-Christophe Raynon SCM Microsystems
Xavier Mariaud STMicroelectronics
Doron Holan Microsoft Corporation
Tzvia Weisman NDS Technologies
Jim Meador STMicroelectronics
Jean-Christophe Lawson Temic Semiconductors
## Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Issue Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8a</td>
<td>July 5, 2000</td>
<td></td>
</tr>
<tr>
<td>0.8b</td>
<td>November 2, 2000</td>
<td>Incorporate RRs 1-12.</td>
</tr>
<tr>
<td>0.9a</td>
<td>Dec 14, 2000</td>
<td>Promotion to 0.9 level</td>
</tr>
<tr>
<td>1.0rc1</td>
<td>Jan 29, 2001</td>
<td>Incorporation of RR 13-16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Proffered as 1.0rc1</td>
</tr>
<tr>
<td>1.00 (release)</td>
<td>March 20, 2001</td>
<td>Device Class ‘0x0B’ Assigned</td>
</tr>
</tbody>
</table>
## Contents

1. **INTRODUCTION** ..................................................................................................................... 7  
   1.1  Related Documents ............................................................................................................. 7  
   1.2  Terms and Abbreviations .................................................................................................... 8  
      - APDU ................................................................................................................................... 8  
      - DI ....................................................................................................................................... 9  
      - WI ...................................................................................................................................... 9  
      - WWT ................................................................................................................................. 10  
   1.3  Document Conventions ..................................................................................................... 10  

2. **OVERVIEW** ......................................................................................................................... 11  

3. **CCID FUNCTIONAL CHARACTERISTICS** ........................................................................ 12  
   3.1  CCID Device Class .......................................................................................................... 12  
      - 3.1.1  Control Commands ................................................................................................. 12  
      - 3.1.2  CCID Events ............................................................................................................ 12  
      - 3.1.3  CCID Command/Response .................................................................................... 13  
   3.2  CCID Device Descriptor .................................................................................................. 14  
   3.3  CCID Configuration Descriptor ...................................................................................... 14  
   3.4  CCID Interface Descriptor .............................................................................................. 14  
   3.5  CCID Class Descriptor .................................................................................................... 15  
   3.6  CCID Endpoints ............................................................................................................... 22  
      - 3.6.1  CCID Command Pipe Bulk-OUT Endpoint ............................................................ 22  
      - 3.6.2  CCID Response Pipe Bulk-IN Endpoint .................................................................. 22  
      - 3.6.3  CCID Event Pipe Interrupt-IN Endpoint ................................................................. 23  
   3.7  CCID Class-Specific Request ......................................................................................... 23  
      - 3.7.1  Abort ....................................................................................................................... 24  
      - 3.7.2  Get Clock Frequencies ............................................................................................ 25  
      - 3.7.3  Get Data Rates ......................................................................................................... 25  

4. **CCID MESSAGES** ................................................................................................................. 27  
   4.1  CCID Command Pipe Bulk-OUT Messages .................................................................... 27
4.1.1 PC_to_RDR_IccPowerOn................................................................................... 28
4.1.2 PC_to_RDR_IccPowerOff.............................................................................. 30
4.1.3 PC_to_RDR_GetSlotStatus......................................................................... 30
4.1.4 PC_to_RDR_XfrBlock............................................................................... 31
4.1.5 PC_to_RDR_GetParameters...................................................................... 33
4.1.6 PC_to_RDR_ResetParameters...................................................................... 34
4.1.7 PC_to_RDR_SetParameters........................................................................ 34
4.1.8 PC_to_RDR_Escape.................................................................................... 37
4.1.9 PC_to_RDR_IccClock................................................................................. 38
4.1.10 PC_to_RDR_T0APDU............................................................................... 39
4.1.11 PC_to_RDR_Secure.................................................................................. 40
4.1.12 PC_to_RDR_Mechanical.......................................................................... 49
4.1.13 PC_to_RDR_Abort................................................................................... 50
4.1.14 PC_to_RDR_SetDataRateAndClockFrequency........................................ 50

4.2 CCID Bulk-IN Messages ............................................................................... 51
4.2.1 Status and Error Reporting in Bulk-IN messages...................................... 51
4.2.2 Bulk-IN message details........................................................................... 52

4.3 CCID Interrupt-IN Messages ...................................................................... 60
4.3.1 RDR_to_PC_NotifySlotChange................................................................. 60
4.3.2 RDR_to_PC_HardwareError.................................................................... 61

5. APPENDIX A SEQUENCES COMMON TO ALL LEVELS OF MESSAGE EXCHANGE.............................................................................................................63

6. APPENDIX B “CHARACTER LEVEL” MESSAGE EXCHANGE SEQUENCES.............................................................................................................67

7. APPENDIX C “APDU LEVEL” MESSAGE EXCHANGE SEQUENCES...........77

8. APPENDIX D “TPDU LEVEL” MESSAGE EXCHANGE SEQUENCES .........89

9. APPENDIX E PIN MANAGEMENT EXAMPLES.................................................97
9.1 PIN Verification .......................................................................................... 97
9.1.1 Example 1: PIN uses a binary format conversion...................................... 97
9.1.2 Example 2: PIN uses a shift rotation format conversion............................ 98
9.1.3 Example 3: PIN uses a BCD format conversion with PIN length insertion... 100
9.1.4 Example 4: PIN uses BCD format conversion with right justification and a control field................................................................................. 102
9.1.5 Example 5: PIN uses an ASCII format conversion with padding............... 104

9.2 PIN Modification....................................................................................... 106
9.2.1 Example 1: Change PIN ASCII format (8-byte long) ........................................ 106
9.2.2 Example 2: PIN uses an ASCII format conversion with PIN length management ................................................................. 108
9.2.3 Character Level, Protocol T=0, sequence for PIN verification ..................... 111

10. APPENDIX G APPLICATION NOTES ................................................................. 113
10.1 Suspend Behavior .......................................................................................... 113

11. APPENDIX H SAMPLES OF CCID DIAGRAMS BASED ON DWFEATURES FIELD ........................................................................................ 115
11.1 dwFeatures fields used in these diagrams ....................................................... 115
11.2 ICC ATRs used in these diagrams ................................................................. 116
11.3 Sample diagrams of voltage management ..................................................... 117
  11.3.1 CCID class AB, ATR1, Feature1 ................................................................. 117
  11.3.2 CCID class B, ATR1, Feature1 ................................................................. 118
  11.3.3 CCID class AB, ATR1, Feature2 (also Feature3, Feature4, Feature5, Feature6) ................................................................. 119
  11.3.4 CCID class AB, ATR2, Feature2 (also Feature3, Feature4, Feature5) ...... 120
11.4 Sample diagrams of rate and protocol management ....................................... 121
  11.4.1 CCID fixed rate (=ATR), ATR 2, Feature1 (or Feature2, Feature3) .............. 121
  11.4.2 CCID with high speed, ATR 4, Feature1 (or Feature2) ................................ 122
  11.4.3 CCID fixed rate (=ATR), ATR 2, Feature4 (or Feature5) ............................ 123
  11.4.4 CCID fixed rate (=ATR), ATR 2, Feature6 .................................................. 124
  11.4.5 CCID fixed rate (=ATR), ATR 3, Feature1 .................................................. 125
  11.4.6 CCID with high speed, ATR 3, Feature1 .................................................... 126
  11.4.7 CCID with high speed, ATR 3, Feature2 (or Feature3) ............................... 127
  11.4.8 CCID with high speed, ATR 3, Feature4 .................................................... 128
  11.4.9 CCID with high speed, ATR 3, Feature5 .................................................... 129
  11.4.10 CCID with high speed, ATR 3, Feature6 ................................................... 130
  11.4.11 CCID high speed, “EMV like”, Cold ATR : ATR1, Warm ATR : ATR4, Feature1 132
11.5 Sample diagrams of automatic IFSD management ........................................ 133
  11.5.1 CCID with large IFSD, ATR4, Feature1 (or Feature2) ............................... 133
  11.5.2 CCID with large IFSD, ATR4, Feature4 ..................................................... 134

12. APPENDIX I ERROR CODES ................................................................. 135
1. Introduction

This document describes proposed requirements and specifications for Universal Serial Bus (USB) devices that interface with or act as interfaces with chip cards and smart cards.

1.1 Related Documents

The following related documents are available from www.usb.org:

- Universal Serial Bus Specification 1.1 (also referred to as the USB specification), September 23, 1998
- Universal Serial Bus Common Class Specification 1.0, December 16, 1997
- Interoperability Specification for ICCs and Personal Computer Systems, Draft Revision 1.0, December 1997

The following related documents can be ordered through www.ansi.org:

- ISO/IEC 7816-2; 1988, Identification Cards – Integrated circuit(s) cards with contacts – Part 2: Dimensions and Locations of the contacts
- ISO/IEC 7816-3; 1997-12-15 Identification Cards – Integrated circuit(s) cards with contacts – Part 3: Electronic signals and transmission protocols
- ISO/IEC 7816-4; 1995 Identification Cards – Integrated circuit(s) cards with contacts – Part 4: Interindustry commands for interchange

The following documents are available from www.emvco.com:

- IFM-EMV 3.1.1; May 31, 1998; EMV ’96 Integrated Circuit Card Specification for Payment Systems;
- IFM-EMV 3.1.1; May 31, 1998; EMV ’96 Integrated Circuit Card Terminal Specification for Payment Systems;
1.2 Terms and Abbreviations

The meanings of some words have been stretched to suit the purposes of this document. These definitions are intended to clarify the discussions that follow. The formulas for BWT, CWT, ETU, and WWT, and the baud rate conversion factor table and clock rate conversion factor table shown below are for reference only. The definitive source for these is the ISO/IEC 7816-3:1997 specification.

**APDU**
Application Protocol Data Unit

**APDU**
The four byte sequence that begins an APDU; CLA INS P1 P2 (ISO 7816-4 5.3.1)

**Command Header**

**ATR**
Answer To Reset

**bps**
Bits per second

**Bps**
Bytes per second

**BWI**
Block Wait Time Integer

**BWT**
Block Waiting time is the maximum delay between the leading edge of the last character of the block received by the ICC and the leading edge of the first character of the next block sent by the ICC for protocol T=1.

\[ BWT = 11 \text{ ETU} + (2^{\text{BWI}} \times 960 \times 372/\text{Clock Frequency}) \]

**CCID**
Chip Card Interface Device

**Chip Card**
used interchangeably with “Smart Card”. See “Smart Card Chapter 9”

**CLA**
Class byte of the command header sent to the ICC.

**Clock Frequency**
The clock frequency currently applied to the ICC.

**Cold RESET**
The sequence described in the ISO/IEC 7816-3:1997 specification section 5.3.2. The sequence starts with the ICC powered off.

**CRC**
Cyclic Redundancy Check

**CWI**
Character Wait time Integer

**CWT**
Character Waiting Time is the maximum delay between the leading edges of two consecutive characters in all blocks for protocol T=1.

\[ CWT = (11 + 2^{\text{CWI}}) \text{ ETU} \]

**Convention**
The convention determines how characters sent to and received from the ICC are interpreted. In direct convention, characters are sent least significant bit first and a “Z” signal state (high) is a ‘1’ bit. In inverse convention, characters are sent most significant bit first and an “A” signal state (low) is a ‘1’ bit.

**D**
Baud rate adjustment factor
DI  Index into Baud rate adjustment factor table

Baud Rate Adjustment Factor Table from ISO/IEC 7816-3:1997

<table>
<thead>
<tr>
<th>DI</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>RFU</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>16</td>
<td>32</td>
<td>RFU</td>
<td>12</td>
<td>20</td>
<td>RFU</td>
<td>RFU</td>
<td>RFU</td>
<td>RFU</td>
<td>RFU</td>
<td>RFU</td>
</tr>
</tbody>
</table>

**dwDefaultClock**  The clock frequency applied by default to the ICC in order to read the ATR data. It is defined in a field in the CCID Class descriptor.

**ETU**  Elementary Time Unit: \( 1 \text{ ETU} = \frac{F}{(D \times \text{Clock Frequency})} \).

**F**  Clock rate adjustment factor

**FI**  Index into Baud rate adjustment factor table

Clock Rate Adjustment Table from ISO/IEC 7816-3:1997

<table>
<thead>
<tr>
<th>FI</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>372</td>
<td>372</td>
<td>558</td>
<td>744</td>
<td>1116</td>
<td>1488</td>
<td>1860</td>
<td>RFU</td>
<td>RFU</td>
<td>512</td>
<td>768</td>
<td>1024</td>
<td>1536</td>
<td>2048</td>
<td>RFU</td>
<td>RFU</td>
</tr>
</tbody>
</table>

**ICC**  Integrated Chip Card

**IFSC**  Information Field Size for ICC for protocol T=1.

**IFSD**  Information Field Size for CCID for protocol T=1.

**INS**  Instruction byte of the command header sent to the ICC.

**ISO/IEC**  International Standards Organization/International Electrotechnical Commission

**Lc**  Optional part of the body of a command APDU. Its size is 1, 2, or 3 bytes. The maximum number of bytes present in this body.

**Le**  Optional part of the body of a command APDU. Its size is 1, 2, or 3 bytes. The maximum number of bytes expected in the data field of the response APDU.

**LRC**  Longitudinal Redundancy Check

**NAD**  Node Address

**P1, P2**  INS parameter of a command header (T=0 or APDU).

**P3**  INS parameter of a T=0 command header. The number of data bytes to be transferred during the command.

**PPS**  Protocol and Parameter Selection

**RFU**  Reserved for Future Use – Must be set to zero unless stated differently.

**Slot**  A physical connection with an ICC

**Smart Card**  Any of a number of similar devices conforming to ISO/IEC 7816 specifications.

**T=0 Command Header**  The sequence of five bytes; CLA INS P1 P2 P3 (ISO 7816-3 8.3.2).

**TPDU**  Transport Protocol Data Unit

**Warm RESET**  The sequence described in the ISO/IEC 7816-3:1997 specification section 5.3.3. The sequence starts with the ICC already powered.

**WI**  Waiting time Integer for protocol T=0
**WWT**

Work Waiting Time is the maximum time allowed between the leading edge of a character sent by the ICC and the leading edge of the previous character sent either by the ICC or the interface device;

\[
\text{WWT} = 960 \times \text{WI} \times \frac{F}{\text{Clock Frequency}}
\]

### 1.3 Document Conventions

Fields that are larger than a byte are stored little endian. Little endian is a method of storing data that places the least significant byte of multiple-byte values at lower storage addresses. For example, a 16-bit integer stored in little endian format places the least significant byte at the lower address and the most significant byte at the next address.

This specification uses the following typographic conventions:

<table>
<thead>
<tr>
<th>Example of convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bValue, bcdName, wOther</td>
<td>Placeholder prefixes such as ‘b’, ‘bcd’, and ‘w’ are used to denote placeholder type. For example: ab array of bytes b bits or bytes dependent on context bcd binary-coded decimal bm bit map w word (2 bytes) dw double word (4 bytes)</td>
</tr>
</tbody>
</table>
2. Overview

Chip cards and Smart cards, as applied to this document, comprise a selection of similar devices conforming to ISO/IEC 7816 specifications. These devices may be used in a wide range of applications. The intent of this document is to specify a protocol by which a host computer may interact with these devices, or with an interface to these devices. Neither the mechanics of the interface, nor the content of the data is of significance in this specification. This document specifies the USB-related configuration information and communication channels.

Generally, when a Chip Card Interface Device is connected to a USB host, it may or may not have a smart card “inserted”. The chip card interface identifies to the host its capabilities and requirements, and the host prepares to communicate with the interface. The interface may then, at any time, detect the presence of a smart card, at which time it communicates that information to the host. As soon as host receives information about the "attached" smart card, further communications may then take place between the host and the smart card through the interface.

The model for the interface defined assumes that a card is or can be inserted into the device. This is the purpose for the “slot change” interrupt message. This model also accommodates devices that integrate CCID and chip card functionality and may be viewed as containing a permanently-inserted ICC.

The remainder of this document specifies the details of these interactions.
3. CCID Functional Characteristics

3.1 CCID Device Class

3.1.1 Control Commands
The CCID class contains one class-specific control-pipe command to abandon/abort the current bulk-pipe command to a slot.
Control-pipe messages are generally used to control a USB device. These messages include standard requests, such as GET_DESCRIPTOR and SET_CONFIGURATION. Commands that are sent on the default pipe report information back to the host on the default pipe. If an error occurs, they generate a standard USB error state. This applies to all class-specific requests in addition to the general requests described in Chapter 9, “USB Device Framework,” in the *Universal Serial Bus Specification*.

![Diagram of HOST and CCID with Control Pipe](image)

3.1.2 CCID Events
Asynchronous events are sent on the interrupt pipe. This allows for events such as a ICC present, ICC removed, or hardware errors such as over current to be sent outside of the standard command/response interface. The Event messages are:

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotifySlotChange</td>
<td>Insertion and removal events.</td>
</tr>
<tr>
<td>HardwareError</td>
<td>A hardware problem was detected on/by the CCID.</td>
</tr>
</tbody>
</table>

![Diagram of HOST and CCID with Interrupt Pipe](image)

The interrupt pipe is mandatory for a CCID that supports ICC insertion/removal. It is optional for a CCID with ICCs that are always inserted and are not removable.
3.1.3 CCID Command/Response

CCID commands are sent on the BULK-OUT endpoint. Each command sent to the CCID has an associated ending response. Some commands can also have intermediate responses. The response is sent on the BULK-IN endpoint. All commands, sent to the specific CCID slot, have to be sent synchronously. A specific slot can accept only one command at a time. A slot is considered to be idle if it is ready to receive a new command.

The Host can send up to \texttt{bMaxCCIDBusySlots} number of commands to the CCID if all of these commands are dedicated to different idle slots. It is the responsibility of the Host to keep track of all busy slots and to not exceed \texttt{bMaxCCIDBusySlots} number of active commands (busy slots). When the number of active slots is equal to \texttt{bMaxCCIDBusySlots}, the CCID shall only accept general requests or the class-specific request to ABORT a slot over the control pipe.

When the CCID successfully receives a new command to a busy slot (not idle), it must fail this command by issuing a \texttt{CMD_SLOT_BUSY} error.

To maintain track of all the commands, the Host sets a unique command identifier \texttt{bSeq} for each command sent to the CCID. To mark command completion the CCID sends the response with the same \texttt{bSeq} number. The driver will not send a new command to a slot until the ending response to the last command to that slot is received. If it is determined that the current command has erred or timed out, the response can be abandoned/aborted by issuing the class specific abandon/abort command to the control endpoint.

More than one BULK-IN message can be sent for each BULK-OUT message. For example, the CCID can send a BULK-IN message with a Time Extension status to notify the host that the ICC has requested more time to process the ICC command and, after a delay, follow this with a second BULK-IN message with the ICC’s response to the command. When this happens, both BULK-IN messages have the same \texttt{bSeq} value.

The CCID is required to send a Zero Length Packet (ZLP) following any Bulk-In message that is a multiple of \texttt{MaxPacketSize}. This ZLP allows the CCID device driver to be more efficient, and is generally considered "good-practice" for USB bulk-in pipes.
3.2 CCID Device Descriptor
The CCID Device descriptor contains no class-specific information. It is a standard, chapter 9 device descriptor. The CCID class is normally specified in a device’s Interface Descriptor.

3.3 CCID Configuration Descriptor
The CCID Configuration descriptor contains no class-specific information. It is a standard, chapter 9 configuration descriptor.

3.4 CCID Interface Descriptor
The CCID Interface descriptor contains the CCID class, subclass, and protocol. The following is a CCID interface descriptor.

Table 3-4. CCID Interface Descriptor

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bLength</td>
<td>1</td>
<td>09h</td>
<td>Size of this descriptor, in bytes.</td>
</tr>
<tr>
<td>1</td>
<td>bDescriptorType</td>
<td>1</td>
<td>04h</td>
<td>INTERFACE descriptor type.</td>
</tr>
<tr>
<td>2</td>
<td>bInterfaceNumber</td>
<td>1</td>
<td>Number</td>
<td>Number of this interface.</td>
</tr>
<tr>
<td>3</td>
<td>bAlternateSetting</td>
<td>1</td>
<td>??h</td>
<td>Alternate setting.</td>
</tr>
<tr>
<td>4</td>
<td>bNumEndpoints</td>
<td>1</td>
<td>02h or 03h</td>
<td>The number of endpoints not including the default. 2 endpoints if the interrupt endpoint is not supported.</td>
</tr>
<tr>
<td>5</td>
<td>bInterfaceClass</td>
<td>1</td>
<td>0Bh</td>
<td>Chip Card Interface Device Class</td>
</tr>
<tr>
<td>6</td>
<td>bInterfaceSubClass</td>
<td>1</td>
<td>00h</td>
<td>Must be 00h. Reserved for future use.</td>
</tr>
<tr>
<td>7</td>
<td>bInterfaceProtocol</td>
<td>1</td>
<td>00h</td>
<td>Must be 00h. Reserved for future use.</td>
</tr>
<tr>
<td>8</td>
<td>iInterface</td>
<td>1</td>
<td>Index</td>
<td>Index of string descriptor for this interface.</td>
</tr>
</tbody>
</table>
3.5 CCID Class Descriptor

The CCID Class includes a class specific descriptor, the CCID Class Descriptor, which specifies certain device features or capabilities as described below.

Table 3-5. CCID Class Descriptor

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bLength</td>
<td>1</td>
<td>36h</td>
<td>Size of this descriptor, in bytes.</td>
</tr>
<tr>
<td>1</td>
<td>bDescriptorType</td>
<td>1</td>
<td>??h</td>
<td>CCID Functional Descriptor type.</td>
</tr>
<tr>
<td>2</td>
<td>bcdCCID</td>
<td>2</td>
<td>0100h</td>
<td>CCID Specification Release Number in Binary-Coded decimal (i.e., 2.10 is 0210h).</td>
</tr>
<tr>
<td>4</td>
<td>bMaxSlotIndex</td>
<td>1</td>
<td></td>
<td>The index of the highest available slot on this device. All slots are consecutive starting at 00h. i.e. 0Fh = 16 slots on this device numbered 00h to 0Fh.</td>
</tr>
<tr>
<td>5</td>
<td>bVoltageSupport</td>
<td>1</td>
<td></td>
<td>This value indicates what voltages the CCID can supply to its slots. The value is a bitwise OR operation performed on the following values: • 01h 5.0V • 02h 3.0V • 04h 1.8V Other bits are RFU.</td>
</tr>
<tr>
<td>6</td>
<td>dwProtocols</td>
<td>4</td>
<td>RRRR</td>
<td>RRRR –Upper Word- is RFU = 0000h PPPP –Lower Word- Encodes the supported protocol types. A ‘1’ in a given bit position indicates support for the associated ISO protocol. 0001h = Protocol T=0 0002h = Protocol T=1 All other bits are reserved and must be set to zero. The field is intended to correspond to the PCSC specification definitions. See PCSC Part3. Table 3-1 Tag 0x0120. Example: 00000003h indicates support for T=0 and T=1.</td>
</tr>
<tr>
<td>10</td>
<td>dwDefaultClock</td>
<td>4</td>
<td></td>
<td>Default ICC clock frequency in KHz encoded as a little endian integer value. Example: 3.58 MHz is encoded as the integer value 3580. (00000DFCh) This is used in ETU and WWT.</td>
</tr>
<tr>
<td>Offset</td>
<td>Field</td>
<td>Size</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------</td>
<td>------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>14</td>
<td>dwMaximumClock</td>
<td>4</td>
<td></td>
<td>Maximum supported ICC clock frequency in KHz encoded as little endian integer value. Example: 14.32 MHz is encoded as the integer value 14320. (000037F0h)</td>
</tr>
<tr>
<td>18</td>
<td>bNumClockSupported</td>
<td>1</td>
<td></td>
<td>The number of clock frequencies that are supported by the CCID. If the CCID does not allow the manual setting of the clock frequency, then this value should be 00h. Otherwise, if the CCID is manual and this value is 00h, the supported clock frequency is assumed to be the default clock frequency defined by dwDefaultClock.</td>
</tr>
<tr>
<td>19</td>
<td>dwDataRate</td>
<td>4</td>
<td></td>
<td>Default ICC I/O data rate in bps encoded as little endian integer Example: 9600 bps is encoded as the integer value 9600. (00002580h)</td>
</tr>
<tr>
<td>23</td>
<td>dwMaxDataRate</td>
<td>4</td>
<td></td>
<td>Maximum supported ICC I/O data rate in bps Example: 115.2Kbps is encoded as the integer value 115200. (0001C200h)</td>
</tr>
<tr>
<td>27</td>
<td>bNumDataRatesSupported</td>
<td>1</td>
<td></td>
<td>The number of data rates that are supported by the CCID. If the CCID does not allow the manual setting of the data rate, then this value should be 00h. Otherwise, if the reader is manual and this value is 00h, the supported data rate is assumed to be the default data rate defined by ISO.</td>
</tr>
<tr>
<td>28</td>
<td>dwMaxIFSD</td>
<td>4</td>
<td></td>
<td>• Indicates the maximum IFSD supported by CCID for protocol T=1.</td>
</tr>
<tr>
<td>32</td>
<td>dwSynchProtocols</td>
<td>4</td>
<td>RRRR</td>
<td>• RRRR-Upper Word- is RFU = 0000h</td>
</tr>
<tr>
<td></td>
<td>PPPP = 000000000h</td>
<td></td>
<td>PPPP-Lower Word- encodes the supported protocol types. A ‘1’ in a given bit position indicates support for the associated protocol. 0001h indicates support for the 2-wire protocol 0002h indicates support for the 3-wire</td>
<td></td>
</tr>
</tbody>
</table>

---

1. This release of the specification does not support devices with the 2-wire, 3-wire, and I2C protocol so PPPP = 0000h. This field is intended to be forward compatible with the PCSC specification.
<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>protocol 1</td>
<td>0004h indicates support for the I2C protocol 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>All other values are outside of this specification, and must be handled by vendor-supplied drivers</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>dwMechanical</td>
<td>4</td>
<td>00000000h, 00000008h</td>
<td>See Note Below. The value is a bitwise OR operation performed on the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 00000000h No special characteristics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 00000001h Card accept mechanism 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 00000002h Card ejection mechanism 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 00000004h Card capture mechanism 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 00000008h Card lock/unlock mechanism</td>
</tr>
<tr>
<td>40</td>
<td>dwFeatures</td>
<td>4</td>
<td></td>
<td>This value indicates what intelligent features the CCID has.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The value is a bitwise OR operation performed on the following values :</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 00000000h No special characteristics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 00000002h Automatic parameter configuration based on ATR data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 00000004h Automatic activation of ICC on inserting</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 00000008h Automatic ICC voltage selection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 00000010h Automatic ICC clock frequency change according to parameters</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 00000020h Automatic baud rate change according to frequency and FI, DI parameters</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 00000040h Automatic parameters negotiation made by</td>
</tr>
</tbody>
</table>

2 These mechanisms of the *dwMechanical* parameter have been included for completeness; however, these functions of motorized CCIDs are not covered by this release of the specification. A future release may attempt to standardize the interface to these mechanical functions.
<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the CCID (use of warm resets, cold resets or PPS according to a manufacturer proprietary algorithm to select the communication parameters with the ICC)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 00000080h Automatic PPS made by the CCID according to the current parameters</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 00000100h CCID can set ICC in clock stop mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 00000200h NAD value other than 00 accepted (T=1 protocol in use)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 00000400h Automatic IFSD exchange as first exchange (T=1 protocol in use)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Only one of the following values may be present:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 00010000h TPDU level exchanges with CCID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 00020000h Short APDU level exchange with CCID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 00040000h Short and Extended APDU level exchange with CCID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Only one of the values 00000040h and 00000080h may be present.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When value 00000040h is present the host shall not try to change the FI, DI, and protocol currently selected.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When an APDU level for exchanges is selected, one of the values 00000040h or 00000080h must be present, as well as the value 00000002h.</td>
</tr>
<tr>
<td>44</td>
<td>dwMaxCCIDMessageLength</td>
<td>4</td>
<td></td>
<td>For extended APDU level the value shall be between 261 + 10 (header) and 65544 +10, otherwise the minimum value is the wMaxPacketSize of the Bulk-OUT endpoint.</td>
</tr>
<tr>
<td>48</td>
<td>bClassGetResponse</td>
<td>1</td>
<td></td>
<td>Significant only for CCID that offers an APDU level for exchanges.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indicates the default class value used by the CCID when it sends a Get Response command to perform the transportation of an APDU by T=0 protocol.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Value FFh indicates that the CCID</td>
</tr>
</tbody>
</table>
### Offset Field Size Value Description

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>49</td>
<td>bClassEnvelope</td>
<td>1</td>
<td></td>
<td>echoes the class of the APDU.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Significant only for CCID that offers an extended APDU level for exchanges.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indicates the default class value used by the CCID when it sends an Envelope command to perform the transportation of an extended APDU by T=0 protocol.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Value FFh indicates that the CCID echoes the class of the APDU.</td>
</tr>
<tr>
<td>50</td>
<td>wLcdLayout</td>
<td>2</td>
<td>XXYYh</td>
<td>Number of lines and characters for the LCD display used to send messages for PIN entry.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>XX: number of lines</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>YY: number of characters per line.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>XXYY=0000h no LCD.</td>
</tr>
<tr>
<td>52</td>
<td>bPINSupport</td>
<td>1</td>
<td>00h-03h</td>
<td>This value indicates what PIN support features the CCID has.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The value is a bitwise OR operation performed on the following values : 01h PIN Verification supported</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>02h PIN Modification supported</td>
</tr>
<tr>
<td>53</td>
<td>bMaxCCIDBusySlots</td>
<td>1</td>
<td>01h – FFh</td>
<td>Maximum number of slots which can be simultaneously busy.</td>
</tr>
</tbody>
</table>

A CCID can announce one of four levels of exchanges with the CCID (Extended APDU, Short APDU, TPDU, or character).

- For APDU level exchanges, the access to the CCID is done using APDU commands and responses as defined in ISO 7816-4. The CCID provides the transportation of the APDU into TPDU. Two APDU levels are defined, short APDU level and extended APDU level, the difference between short APDU and extended APDU are explained in 5.3.2 of ISO 7816-4. A CCID that indicates a short APDU level only accepts short APDU. A CCID that indicates an extended APDU level accepts both short APDU and extended APDU.

If the ICC requests time extension, by using a NULL procedure byte (60h) in T=0 protocol or S(WTX) in T=1 protocol, the CCID informs the host of this request.

As a CCID at an APDU level provides a high level of automatism in ICC exchange, it shall also provide a high level of automatism in ATR treatment and shall implement one of the automatism : automatic parameters negotiation made by the CCID (proprietary algorithm), or automatic PPS made by the CCID according to the current parameters. At least two standards of transportation for APDU are defined, ISO 7816-
4 and EMV 3.1.1, the choice of standard to implement is out of the scope of this specification.

- For TPDU level exchanges, the access to the CCID is done using TPDU. The TPDU format changes according to the protocol or for PPS exchange.

TPDU for PPS exchange has the format:

Command TPDU : FF PPS0 PPS1 PPS2 PPS3 PCK, with PPS1, PPS2, PPS3 optional (see 7816-3 chapter 7)
Response TPDU : FF PPS0_R PPS1_R PPS2_R PPS3_R PCK_R, with PPS1_R, PPS2_R, PPS3_R optional (see 7816-3 7.4)

The CCID implements and verifies timings and protocol according to its parameters settings to assume ISO 7816-3 7.1, 7.2 (“initial waiting time”, …). But no check on frame format is mandatory on request, and on response the only recommended analysis is the most significant nibble of PPS0_R to compute the number of bytes left to receive.

A CCID that implements automatic PPS should not accept TPDU for PPS exchange and must check for PPS response validity.

T=0 TPDU can have three formats (see ISO 7816-3, chapter 8.3.2):
- Form 1, no data to exchange with ICC, only header:
  Command TPDU = CLA INS P1 P2, the CCID is responsible to add P3=00h.
  Response TPDU = SW1 SW2
- Form 2, data expected from ICC:
  Command TPDU = CLA INS P1 P2 Le, Le=P3 from 00h to FFh (00h means 100h)
  Response TPDU = Data(Le) SW1 SW2, Data(Le) is for the Le data received from the ICC or empty if ICC rejects the command.
- Form 3, data are to be sent to the ICC:
  Command TPDU = CLA INS P1 P2 Lc Data(Lc), Lc=P3 from 01h to FFh and Data(Lc) for the Lc data to send to the ICC.
  Response TPDU = SW1 SW2

The CCID, for T=0 TPDU, is in charge of managing procedure bytes (ISO 7816-3 8.3.3) and character level (ISO 7816-3 8.2). The procedure bytes are not mapped into the response TPDU except for the SW1 SW2 bytes.

The CCID implements and verifies timings according to its parameters settings to assume ISO 7816-3 8.2 (work waiting time, extra guard time, …).

If ICC uses NULL procedure byte (60h) the CCID informs the host of this request for time extension.

T=1 TPDU command and response use the frame format described in 7816-3 chapter 9.4 for T=1 protocol. The CCID expects the respect of the character frame of ISO 7816-3 9.4.1. But no check on frame format is mandatory on sending, and on receiving the only recommended checks are:
- Expecting LEN byte as third byte.
- Waiting for LEN bytes as INF field.
- Wait for an EDC field which length complies with parameter bmTCCKST1 (see command PC_to_RDR_SetParameters).

The CCID implements and verifies timing according to its parameters settings to assume ISO 7816-3 9.5.3 (CWT, BWT, BGT, ...).

The detection of parity error on character received is optional.
The interpretation of first bytes received as NAD and PCB to manage VPP is optional and depends on CCID capabilities.

• For character level exchange, the CCID provides only the management of character level, including timings, as defined in ISO 7816-3 8.2 for T=0 and in ISO 7816-3 9.3 for T=1.

The CCID sends the characters in the command (maybe none) then waits for the number of characters (if not null) indicated in the command.

To insure timing respects the CCID uses the current parameters.

Note: For Parameter dwFeatures:
• If no level for exchanges with ICC is indicated the level is character exchange. The CCID implements the character frame and character repetition procedure (if T=0 is selected).
• When a CCID doesn't declare the values 00000010h and 00000020h, the frequency or the baud rate must be made via manufacturer proprietary PC_to_RDR_Escape command.
• When a CCID using TPDU exchange level declares neither of the values 00000040h, or 00000080h, the PPS exchange must be made using a TPDU for PPS exchange.

Note: If the CCID does not support queuing, it reports bMaxCCIDBusySlots equal to 1; which means that the CCID will treat all commands synchronously.
3.6 CCID Endpoints

A Chip Card Interface Device (CCID) shall support a minimum of two endpoints in addition to the default (control) endpoint: one bulk-out and one bulk-in. A CCID that reports ICC insertion or removal events must also support an interrupt endpoint.

<table>
<thead>
<tr>
<th>Interrupt</th>
<th>IN</th>
<th>Optional</th>
<th>CCID Event Notification pipe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulk OUT</td>
<td>OUT</td>
<td>Mandatory</td>
<td>CCID Command pipe</td>
</tr>
<tr>
<td>Bulk IN</td>
<td>IN</td>
<td>Mandatory</td>
<td>CCID Response pipe</td>
</tr>
</tbody>
</table>

3.6.1 CCID Command Pipe Bulk-OUT Endpoint

The Bulk Out Endpoint is used to send commands and transfer data from the host to the device.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bLength</td>
<td>1</td>
<td>07h</td>
<td>Size of this descriptor in bytes</td>
</tr>
<tr>
<td>1</td>
<td>bDescriptorType</td>
<td>1</td>
<td>05h</td>
<td>ENDPOINT descriptor type</td>
</tr>
<tr>
<td>2</td>
<td>bEndpointAddress</td>
<td>1</td>
<td>01-0Fh</td>
<td>The address of this endpoint on the USB device. This address is an endpoint number between 1 and 15. Bit 0..3 Endpoint number Bit 4..6 Reserved, must be 0 Bit 7 0 = Out</td>
</tr>
<tr>
<td>3</td>
<td>bmAttributes</td>
<td>1</td>
<td>02h</td>
<td>This is a Bulk endpoint</td>
</tr>
<tr>
<td>4</td>
<td>wMaxPacketSize</td>
<td>2</td>
<td>00??h</td>
<td>Maximum data transfer size can be 8,16,32, or 64 bytes</td>
</tr>
<tr>
<td>6</td>
<td>bInterval</td>
<td>1</td>
<td>00h</td>
<td>Does not apply to Bulk endpoints</td>
</tr>
</tbody>
</table>

3.6.2 CCID Response Pipe Bulk-IN Endpoint

The Bulk In endpoint is used to send responses and transfer data from the device to the host in reply to commands received on the Command Pipe.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bLength</td>
<td>1</td>
<td>07h</td>
<td>Size of this descriptor in bytes</td>
</tr>
<tr>
<td>1</td>
<td>bDescriptorType</td>
<td>1</td>
<td>05h</td>
<td>ENDPOINT descriptor type</td>
</tr>
<tr>
<td>2</td>
<td>bEndpointAddress</td>
<td>1</td>
<td>81-8Fh</td>
<td>The address of this endpoint on the USB device. This address is an endpoint number between 1 and 15. Bit 0..3 Endpoint number Bit 4..6 Reserved, must be 0 Bit 7 1 = In</td>
</tr>
<tr>
<td>3</td>
<td>bmAttributes</td>
<td>1</td>
<td>02h</td>
<td>This is a Bulk endpoint</td>
</tr>
<tr>
<td>4</td>
<td>wMaxPacketSize</td>
<td>2</td>
<td>00??h</td>
<td>Maximum data transfer size can be</td>
</tr>
</tbody>
</table>
3.6.3 CCID Event Pipe Interrupt-IN Endpoint

The Event Report pipe is used by the CCID to notify the host of an ICC insertion event, ICC removal event, or hardware errors such as over current.

The interrupt pipe is mandatory for a CCID that supports ICC insertion/removal. It is optional for a CCID with ICCs that are always inserted and are not removable. If there is an Interrupt-In endpoint, then the RDR_to_PC_NotifySlotChange message is required and all other messages are optional.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bLength</td>
<td>1</td>
<td>07h</td>
<td>Size of this descriptor in bytes</td>
</tr>
<tr>
<td>1</td>
<td>bDescriptorType</td>
<td>1</td>
<td>05h</td>
<td>ENDPOINT descriptor type</td>
</tr>
<tr>
<td>2</td>
<td>bEndpointAddress</td>
<td>1</td>
<td>81-8Fh</td>
<td>The address of this endpoint on the USB device. This address is an endpoint number between 1 and 15. It must be different from the Bulk-IN endpoint address. Bit 0..3 Endpoint number Bit 4..6 Reserved, must be 0 Bit 7 1 = In</td>
</tr>
<tr>
<td>3</td>
<td>bmAttributes</td>
<td>1</td>
<td>03h</td>
<td>This is an Interrupt endpoint</td>
</tr>
<tr>
<td>4</td>
<td>wMaxPacketSize</td>
<td>2</td>
<td>00??h</td>
<td>Maximum data transfer size (depends on the size of the RDR_to_PC_NotifySlotChange message but it is at least 2h)</td>
</tr>
<tr>
<td>6</td>
<td>bInterval</td>
<td>1</td>
<td>??h</td>
<td>Interval for polling endpoint for data transfers. Expressed in milliseconds. Must be in the range from 1 to 255. The recommended value is 255.</td>
</tr>
</tbody>
</table>

3.7 CCID Class-Specific Request

Three class-specific requests are defined. The first request allows the host to abort the response portion of a command/response message pair. This may be necessary to recover from error conditions and put the CCID into a state where it can receive a new command message. The two others requests allow the CCID that does not automatically determine the clock frequency or the baud rate according to the TA1 procedure byte, to indicate to the host the clock frequencies or the data rates it is able to support. These requests are
done through the default pipe (control endpoint) and therefore follow the format of the
default pipe requests as defined in the Universal Serial Bus specification.

The following table defines valid values of bRequest.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01h</td>
<td>ABORT</td>
</tr>
<tr>
<td>02h</td>
<td>GET_CLOCK_FREQUENCIES</td>
</tr>
<tr>
<td>03h</td>
<td>GET_DATA_RATES</td>
</tr>
</tbody>
</table>

### 3.7.1 Abort

This Control pipe request works with the Bulk-OUT PC_to_RDR_Abort command to tell
the CCID to stop any current transfer at the specified slot and return to a state where the
slot is ready to accept a new command pipe Bulk-OUT message. The wValue field
contains the slot number (bSlot) in the low byte and the sequence number (bSeq) in the
high byte. The bSlot value tells the CCID which slot should be aborted. The bSeq number
is not related to the slot number or the Bulk-OUT message being aborted, but it does
match bSeq in the PC_to_RDR_Abort command.

When the Host wants to abort a Bulk-OUT command message, it will send a pair of abort
messages to the CCID. Both will have the same bSlot and bSeq values. The Host will
first send this request to the CCID over the Control pipe. The Host will next send the
PC_to_RDR_Abort command message over the Bulk-OUT pipe. Both are necessary due
to the asynchronous nature of control pipes and Bulk-Out pipes relative to each other.

Upon receiving the Control pipe ABORT request the CCID should check the state of the
requested slot. If the last Bulk-OUT message received by the CCID was a
PC_to_RDR_Abort command with the same bSlot and bSeq as the ABORT request, then
the CCID will respond to the Bulk-OUT message with the RDR_to_PC_SlotStatus
response.

If the previous Bulk-OUT message received by the CCID was not a PC_to_RDR_Abort
command with the same bSlot and bSeq as the ABORT request, then the CCID will fail
all Bulk-Out commands to that slot until the PC_to_RDR_Abort command with the same
bSlot and bSeq is received. Bulk-OUT commands will be failed by sending a response
with bmCommandStatus=Failed and bError=CMD_ABORTED.

When the CCID has received both the Control pipe ABORT request and the Bulk-OUT
PC_to_RDR_Abort command, it will send the RDR_to_PC_SlotStatus Bulk-IN reply
with good status on the response pipe (see 4.2 CCID response Bulk-IN messages).

If the CCID receives the ABORT request and a response has been partially sent, the
CCID shall accept the request, finish sending the response or send an "interrupted" partial
response, and fail all Bulk-Out commands to that slot until the PC_to_RDR_Abort command with the same bSlot and bSeq is received.

It is the responsibility of the Host to keep track of pending ABORT commands for slots and discard all responses from the aborted slots, if any, until the Host receives the RDR_to_PC_SlotStatus which matches the PC_to_RDR_Abort message. The Host can send a new command for the slot only after receiving the RDR_to_PC_SlotStatus Bulk-IN response from the CCID for the specified slot.

<table>
<thead>
<tr>
<th>bmRequestType</th>
<th>bRequest</th>
<th>wValue</th>
<th>wIndex</th>
<th>wLength</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>00100001B</td>
<td>ABORT</td>
<td>bSeq,bSlot</td>
<td>Interface</td>
<td>0000h</td>
<td>None</td>
</tr>
</tbody>
</table>

### 3.7.2 Get Clock Frequencies

This requests the CCID to report a list of selectable clock frequencies which the CCID is capable of supporting. The response is an array of double-words. The clock frequencies are reported in KHz encoded as a little endian integer value.

A CCID which implements automatic clock frequency selection (bit 10h, in dwFeatures in the class descriptor) does not have to support this request.

\[
\text{wLength} = \text{bNumClockSupported} \times \text{sizeof(double word)}.
\]

<table>
<thead>
<tr>
<th>bmRequestType</th>
<th>bRequest</th>
<th>wValue</th>
<th>wIndex</th>
<th>wLength</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>10100001B</td>
<td>GET_CLOCK_FREQ UENCIES</td>
<td>0000h</td>
<td>Interface</td>
<td>length of data</td>
<td>array of double words</td>
</tr>
</tbody>
</table>

### 3.7.3 Get Data Rates

This requests the CCID to report a list of selectable data rates which the CCID is capable of supporting. The response is an array of double-words. The data rates are reported in bps encoded as a little endian integer value.

A CCID which implements automatic data rate selection (bit 20h, in dwFeatures in the class descriptor) does not have to support this request.

\[
\text{wLength} = \text{bNumDataRatesSupported} \times \text{sizeof(double word)}.
\]
<table>
<thead>
<tr>
<th>bmRequestType</th>
<th>bRequest</th>
<th>wValue</th>
<th>wIndex</th>
<th>wLength</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>10100001B</td>
<td>GET_DATA_RATES</td>
<td>0000h</td>
<td>Interface</td>
<td>length of data</td>
<td>array of double words</td>
</tr>
</tbody>
</table>
4. CCID Messages

All bulk messages begin with a 10-byte header, followed by message-specific data. The header consists of a message type (1 byte), a length field (four bytes), the slot number (1 byte), a sequence number field (1 byte), and either three message specific bytes, or a status field (1 byte), an error field and one message specific byte. The purpose of the 10-byte header is to provide a constant offset at which message data begins across all messages.

A Bulk-OUT message is a command, and always receives at least one Bulk-IN message in response. The response messages always contains the exact same slot number, and sequence number fields from the header that was contained in the Bulk-OUT command message.

The message type (bMessageType) identifies the message. The length field (dwLength) is the length of the message not including the 10-byte header.

The slot number (bSlot) identifies which ICC slot is being addressed by the message, if the CCID supports multiple slots. The slot number is zero-relative, and is in the range of zero to FFh.

The sequence number (bSeq) is a monotonically increasing by one counter of bulk messages sent to the CCID. Because the response to a command always uses the exact same sequence number contained in the command, the host can use the sequence number in a response message to verify that a particular response is the one expected in reply to a particular command. This sequence number is not related to any interaction between the CCID and the ICC itself, but simply tracks the USB bulk message exchanges between the host and the CCID. The initial value of the sequence number is not important, but typically starts at zero.

Slot Status (bStatus) is returned in the Bulk-IN message response.

Slot Error (bError) is returned in the Bulk-IN message response.

The remaining bytes of the header (3 bytes in Bulk-OUT messages and 1 byte in Bulk-IN message) are message specific.

### 4.1 CCID Command Pipe Bulk-OUT Messages

<table>
<thead>
<tr>
<th>Message Name</th>
<th>bMessageType</th>
<th>Section</th>
<th>Response Message(s)</th>
<th>Abortable</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC to RDR IccPowerOn</td>
<td>62h</td>
<td>4.1.1</td>
<td>RDR to PC DataBlock</td>
<td>Yes</td>
</tr>
<tr>
<td>PC to RDR IccPowerOff</td>
<td>63h</td>
<td>4.1.2</td>
<td>RDR to PC SlotStatus</td>
<td>No</td>
</tr>
<tr>
<td>PC to RDR GetSlotStatus</td>
<td>65h</td>
<td>4.1.3</td>
<td>RDR to PC SlotStatus</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 4-1. Summary of Bulk-Out Messages
<table>
<thead>
<tr>
<th>Message Name</th>
<th>bMessageType</th>
<th>Section</th>
<th>Response Message(s)</th>
<th>Abortable</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC_to_RDR_XfrBlock</td>
<td>6Fh</td>
<td>4.1.4</td>
<td>RDR_to_PC_DataBlock</td>
<td>Yes</td>
</tr>
<tr>
<td>PC_to_RDR_GetParameters</td>
<td>6Ch</td>
<td>4.1.5</td>
<td>RDR_to_PC_Parameters</td>
<td>No</td>
</tr>
<tr>
<td>PC_to_RDR_ResetParameters</td>
<td>6Dh</td>
<td>4.1.6</td>
<td>RDR_to_PC_Parameters</td>
<td>No</td>
</tr>
<tr>
<td>PC_to_RDR_SetParameters</td>
<td>61h</td>
<td>4.1.7</td>
<td>RDR_to_PC_Parameters</td>
<td>No</td>
</tr>
<tr>
<td>PC_to_RDR_Escape</td>
<td>6Bh</td>
<td>4.1.8</td>
<td>RDR_to_PC_Escape</td>
<td>Yes</td>
</tr>
<tr>
<td>PC_to_RDR_IccClock</td>
<td>6Eh</td>
<td>4.1.9</td>
<td>RDR_to_PC_SlotStatus</td>
<td>No</td>
</tr>
<tr>
<td>PC_to_RDR_T0APDU</td>
<td>6Ah</td>
<td>4.1.10</td>
<td>RDR_to_PC_SlotStatus</td>
<td>No</td>
</tr>
<tr>
<td>PC_to_RDR_SetDataRateAndClockFrequency</td>
<td>73h</td>
<td>4.1.14</td>
<td>RDR_to_PC_DataRateAndClockFrequency</td>
<td>No</td>
</tr>
</tbody>
</table>

The fields bmICCStatus and BmCommandStatus used in the following error tables are defined in 4.2.1.

### 4.1.1 PC_to_RDR_IccPowerOn

When an ICC has been removed from a slot or after the processing of a message leads the CCID to power off an ICC in a slot, that slot’s power and contacts will be “inactive”.

A PC_to_RDR_IccPowerOn message to an inactive slot will return an Answer To Reset (ATR) data. If the response did not have an error, then the slot will be left “active” and the ATR returned could be the cold ATR or a warm ATR depending on the automatism the CCID implements. If the response did have an error, then the slot will be left inactive. Because each warm reset to an ICC can return a sequentially significant warm reset ATR, any subsequent PC_to_RDR_IccPowerOn message sent to an active slot will return the next warm reset ATR.

When an ICC is inserted into a slot in a CCID which features “automatic activation of ICC on inserting”, the CCID can notify the host of an ICC insertion and proceed to activate the ICC, beginning with a cold reset ATR. If the CCID also features “automatic ICC voltage selection” and the first voltage applied was incorrect, the CCID will power off the ICC, change the voltage and go through the standard reset sequence again. When the final voltage is reached the CCID then will proceed to other automatism it implements, if any, to get an ATR. The CCID will respond to the first PC_to_RDR_IccPowerOn message it receives for that slot with the final ATR it got, independent of where the CCID may have been in the voltage selection and automatism sequence when it received the message. For a CCID which features “automatic activation of ICC on inserting”, the bPowerSelect field must be 00h (Automatic Voltage Selection) in the first PC_to_RDR_IccPowerOn message received for a slot which was previously inactive.

A CCID which does not feature “automatic activation of ICC on inserting”, will notify the host of an ICC insertion and wait, with its contacts inactive, for the
PC_to_RDR_IccPowerOn message before doing the standard reset sequence at the voltage specified by the bPowerSelect field. If the CCID features “automatic ICC voltage selection” and the first PC_to_RDR_IccPowerOn message selects automatic voltage selection in the bPowerSelect field, the CCID will return an ATR of the final voltage. To change the voltage on a slot that is active, the slot must first be powered off before issuing a PC_to_RDR_IccPowerOn message with a new bPowerSelect value. For reference, ISO/IEC 7816-3:1997 section 4.2.2 requires that the power must be off for at least 10 milliseconds.

On reception of a PC_to_RDR_IccPowerOn command, the CCID automatically resets the parameters to their default values except for the following case. If the CCID has the "automatic activation of ICC on inserting" feature, and the PC_to_RDR_IccPowerOn command is the first PC_to_RDR_IccPowerOn command the CCID has received since either 1) the ICC was first inserted into the slot or 2) the CCID has resumed from the suspended state, then the CCID will not reset the parameters because the CCID already reset them before it started the automatic activation sequence. The "automatic activation" CCID keeps the parameters it read and negotiated during the automatic activation sequence. On reception of any subsequent PC_to_RDR_IccPowerOn commands, the "automatic activation" CCID automatically resets the parameters to their default values.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>62h</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>dwLength</td>
<td>4</td>
<td>0000000000h</td>
<td>Size of extra bytes of this message</td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td>00-FFh</td>
<td>Identifies the slot number for this command</td>
</tr>
<tr>
<td>6</td>
<td>bSeq</td>
<td>1</td>
<td>00-FFh</td>
<td>Sequence number for command.</td>
</tr>
<tr>
<td>7</td>
<td>bPowerSelect</td>
<td>1</td>
<td>00h, 01h, 02h, or 03h</td>
<td>Voltage that is applied to the ICC 00h – Automatic Voltage Selection 01h – 5.0 volts 02h – 3.0 volts 03h – 1.8 volts</td>
</tr>
<tr>
<td>8</td>
<td>abRFU</td>
<td>2</td>
<td></td>
<td>Reserved for Future Use</td>
</tr>
</tbody>
</table>

The response to this message is the RDR_to_PC_DataBlock message and the data returned is the Answer To Reset (ATR) data. This message could return any of the following errors.

<table>
<thead>
<tr>
<th>bmICC</th>
<th>bmCommand</th>
<th>bError</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>bSlot does not exist</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>ICC_MUTE</td>
<td>No ICC present</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>HW_ERROR</td>
<td>Hardware error</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>7</td>
<td>bPowerSelect error (not supported)</td>
</tr>
</tbody>
</table>
### 4.1.2 PC_to_RDR_IccPowerOff

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>63h</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>dwLength</td>
<td>4</td>
<td>0000000000h</td>
<td>Size of extra bytes of this message</td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td>00-FFh</td>
<td>Identifies the slot number for this command</td>
</tr>
<tr>
<td>6</td>
<td>bSeq</td>
<td>1</td>
<td>00-FFh</td>
<td>Sequence number for command.</td>
</tr>
<tr>
<td>7</td>
<td>abRFU</td>
<td>3</td>
<td></td>
<td>Reserved for Future Use</td>
</tr>
</tbody>
</table>

The response to this message is the RDR_to_PC_SlotStatus message.
This message could return any of the following errors.

<table>
<thead>
<tr>
<th>bmICC Status</th>
<th>bmCommand Status</th>
<th>bError</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>bSlot does not exist</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>BUSY_WITH_AUTO_SEQUENCE</td>
<td>Automatic sequence on-going</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>Command Not Supported</td>
</tr>
</tbody>
</table>
| 0,1,2        | 1               | CMD_SLOT_BUSY             |}

### 4.1.3 PC_to_RDR_GetSlotStatus

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>65h</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>dwLength</td>
<td>4</td>
<td>0000000000h</td>
<td>Size of extra bytes of this message</td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td>00-FFh</td>
<td>Identifies the slot number for this command</td>
</tr>
</tbody>
</table>

4. CCID Messages

Page 30 of 135

20 Mar 01 @ 8:53
The response to this message is the RDR_to_PC_SlotStatus message. This message could return any of the following errors.

<table>
<thead>
<tr>
<th>bmICC Status</th>
<th>bmCommand Status</th>
<th>bError</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>bSlot does not exist</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>ICC_MUTE</td>
<td>No ICC present</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>HW_ERROR</td>
<td>Hardware error</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Command Not Supported</td>
</tr>
<tr>
<td>0,1,2</td>
<td>1</td>
<td>CMD_SLOT_BUSY</td>
<td></td>
</tr>
</tbody>
</table>

4.1.4 PC_to_RDR_XfrBlock

The block should never exceed the dwMaxCCIDMessageLength-10 in the Class Descriptor.

Note: For reference, the absolute maximum block size for a TPDU T=0 block is 260 bytes (5 bytes command; 255 bytes data), or for a TPDU T=1 block is 259 bytes, or for a short APDU T=1 block is 261 bytes, or for an extended APDU T=1 block is 65544 bytes.
### Offset Field Size Value Description

- TPDU level, RFU, = 0000h
- short APDU level, RFU, = 00000h
- extended APDU level, indicates if APDU begins or ends in this command:
  - 0000h the APDU command begins and ends with this command,
  - 0001h the APDU command begins with this command, and continue in next PC_to_RDR_XfrBlock,
  - 0002h this abData field continues an APDU command and ends the APDU command,
  - 0003h the abData field continues an APDU command and another block is to follow,
  - 0010h empty abData field, continuation of APDU response is expected in the next RDR_to_PC_DataBlock.

### 10 abData Byte array

Data block sent to the CCID. Depending on the exchange level, the CCID may send this data “as is” to the ICC, or may modify it before sending it to the ICC. (0 to 65544 bytes)

Parameter $bBWI$ is only used by CCIDs which use the character level and TPDU level of exchange (as reported in the $dwFeatures$ parameter in the CCID Functional Descriptor) and only for protocol $T=1$ transfers.

**The response to this message is the RDR_to_PC_DataBlock message.**

This message could return any of the following errors.

<table>
<thead>
<tr>
<th>bmICC Status</th>
<th>bmCommand Status</th>
<th>bError</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>bSlot does not exist</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>ICC_MUTE</td>
<td>No ICC present</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>BUSY_WITH_AUTO_SEQUENCE</td>
<td>Automatic sequence on-going</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>HW_ERROR</td>
<td>Hardware error</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Command Not Supported</td>
</tr>
<tr>
<td>0,1,2</td>
<td>1</td>
<td>CMD_SLOT_BUSY</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>7</td>
<td>bPowerselect error (not supported)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>XFR_PARITY_ERROR</td>
<td>parity error</td>
</tr>
</tbody>
</table>
### 4.1.5 PC_to_RDR_GetParameters

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>6Ch</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>dwLength</td>
<td>4</td>
<td>00000000h</td>
<td>Size of extra bytes of this message</td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td>00-FFh</td>
<td>Identifies the slot number for this command</td>
</tr>
<tr>
<td>6</td>
<td>bSeq</td>
<td>1</td>
<td>00-FFh</td>
<td>Sequence number for command.</td>
</tr>
<tr>
<td>7</td>
<td>abRFU</td>
<td>3</td>
<td></td>
<td>Reserved for Future Use</td>
</tr>
</tbody>
</table>

The response to this message is the RDR_to_PC_Parameters message.

This message could return any of the following errors.

<table>
<thead>
<tr>
<th>bmICC Status</th>
<th>bmCommand Status</th>
<th>bError</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>bSlot does not exist</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>ICC_MUTE</td>
<td>No ICC present</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>BUSY_WITH_AUTO_SEQUENCE</td>
<td>Automatic sequence on-going</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>HW_ERROR</td>
<td>Hardware error</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Command Not Supported</td>
</tr>
<tr>
<td>0,1,2</td>
<td>1</td>
<td>CMD_SLOT_BUSY</td>
<td></td>
</tr>
</tbody>
</table>
### 4.1.6 PC_to_RDR_ResetParameters

This command resets the slot parameters to their default values.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>6Dh</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>dwLength</td>
<td>4</td>
<td>00000000h</td>
<td>Size of extra bytes of this message</td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td>00-FFh</td>
<td>Identifies the slot number for this command</td>
</tr>
<tr>
<td>6</td>
<td>bSeq</td>
<td>1</td>
<td>00-FFh</td>
<td>Sequence number for command. Note – not related to the slot number and rolls over to 00h after FFh.</td>
</tr>
<tr>
<td>7</td>
<td>abRFU</td>
<td>3</td>
<td></td>
<td>Reserved for Future Use</td>
</tr>
</tbody>
</table>

#### The response to this message is the RDR_to_PC_Parameters message.

This message could return any of the following errors.

<table>
<thead>
<tr>
<th>bmICC Status</th>
<th>bmCommand Status</th>
<th>bError</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>bSlot does not exist</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>No ICC present</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>BUSY_WITH_AUTO_SEQUENCE</td>
<td>Automatic sequence on-going</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>HW_ERROR</td>
<td>Hardware error</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Command Not Supported</td>
</tr>
<tr>
<td>0,1,2</td>
<td>1</td>
<td>CMD_SLOT_BUSY</td>
<td></td>
</tr>
</tbody>
</table>

### 4.1.7 PC_to_RDR_SetParameters

This command is used to change the parameters for a given slot.

A CCID which has no automatic features (dwFeatures=0, 100h, 200h, or 300h) depends on the driver to send this command to set the protocol and other parameters to the right values necessary to correctly talk to the ICC located in the selected slot.

A CCID which has automatic features will automatically set the protocol and certain parameters based on data received from the ICC (ATR, PPS, IFSD, or proprietary algorithms). The level of automatism and design requirements will determine which parameters the CCID will allow the driver to change.

If this command tries to change a parameter which is not changeable, then the CCID will not change any parameters and the RDR_to_PC_GetParameters response will return a Command Failed status and the bError field will contain the offset of the "offending" parameter.
### CCID Messages

#### Offset Field Size Value Description

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>61h</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>dwLength</td>
<td>4</td>
<td></td>
<td>Size of extra bytes of this message</td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td>00-FFh</td>
<td>Identifies the slot number for this command</td>
</tr>
<tr>
<td>6</td>
<td>bSeq</td>
<td>1</td>
<td>00-FFh</td>
<td>Sequence number for command.</td>
</tr>
</tbody>
</table>
| 7      | bProtocolNum       | 1    | 00h, 01h| Specifies what protocol data structure follows.  
|        |                    |      | 00h = Structure for protocol T=0  
|        |                    |      | 01h = Structure for protocol T=1  
|        |                    |      | The following values are reserved for future use.  
|        |                    |      | 80h = Structure for 2-wire protocol  
|        |                    |      | 81h = Structure for 3-wire protocol  
|        |                    |      | 82h = Structure for I2C protocol  |
| 8      | abRFU              | 2    |       | Reserved for Future Use                                                    |
| 10     | abProtocolDataStructure | Byte array | Protocol Data Structure        |

#### Protocol Data Structure for Protocol T=0 (bProtocolNum=0) (dwLength=00000005h)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| 10     | bmIndexDindex      | 1    |       | B7-4 – FI – Index into the table 7 in ISO/IEC 7816-3:1997 selecting a clock rate conversion factor  
|        |                    |      |       | B3-0 – DI - Index into the table 8 in ISO/IEC 7816-3:1997 selecting a baud rate conversion factor |
| 11     | bmTCCKST0          | 1    | 00h, 02h| For T=0, B0 = 0b, B7-2 = 0000000b  
|        |                    |      |       | B1 – Convention used (b1=0 for direct, b1=1 for inverse)  
|        |                    |      |       | Note: The CCID ignores this bit. Its value is determined by the first byte of the ICC’s ATR data. It is here as a placeholder so the same data structure can be used for the PC_to_RDR_SetParameters and the RDR_to_PC_GetParameters messages  
|        |                    |      |       | This field is intended to be compatible with parameter rr in Table 2-6 of Part 4 of the PCSC specification. |
| 12     | bGuardTimeT0       | 1    | 00-FFh| Extra Guardtime between two characters. Add 0 to 254 etu to the normal guardtime of 12etu. FFh is the same as 00h. |
| 13     | bWaitingIntegerT0  | 1    | 00-FFh| W1 for T=0 used to define WWT                                                  |
Protocol Data Structure for Protocol T=1 (bProtocolNum=1) (dwLength=00000007h)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| 10     | bmFindexDindex      | 1    |       | B7-4 – FI – Index into the table 7 in ISO/IEC 7816-3:1997 selecting a clock rate conversion factor  
        |                     |      |       | B3-0 – DI - Index into the table 8 in ISO/IEC 7816-3:1997 selecting a baud rate conversion factor |
| 11     | bmTCCKST1           | 1    | 10h, 11h, 12h, 13h | For T=1, B7-2 – 000100b  
        |                     |      |       | B0 – Checksum type (b0=0 for LRC, b0=1 for CRC  
        |                     |      |       | B1 – Convention used (b1=0 for direct, b1=1 for inverse)  
        |                     |      |       | Note: The CCID ignores this bit. Its value is determined by the first byte of the ICC’s ATR data. It is here as a placeholder so the same data structure can be used for the PC_to_RDR_SetParameters and the RDR_to_PC_GetParameters messages. This field is intended to be compatible with parameter rr in Table 2-6 of Part 4 of the PCSC specification. |
| 12     | bGuardTimeT1        | 1    | 00-FFh | Extra Guardtime (0 to 254 etu between two characters). If value is FFh, then guardtime is reduced by 1 etu. |
| 13     | bmWaitingIntegersT1 | 1    | 00-9Fh | B7-4 = BWI values 0-9 valid  
        |                     |      |       | B3-0 = CWI values 0-Fh valid |
| 14     | bClockStop          | 1    | 00-03h | ICC Clock Stop Support  
        |                     |      |       | 00 = Stopping the Clock is not allowed  
        |                     |      |       | 01 = Stop with Clock signal Low  
        |                     |      |       | 02 = Stop with Clock signal High  
        |                     |      |       | 03 = Stop with Clock either High or Low |
| 15     | bIFSC               | 1    | 00-FEh | Size of negotiated IFSC |
| 16     | bNadValue           | 1    |       | Value = 00h if CCID doesn't support a value other than the default value. Else value respects ISO/IEC 7816-3, |
4. CCID Messages

Protocol Data Structures for “2-wire” protocol (bProtocolNum=80h), “3-wire” protocol (bProtocolNum=81h), and “I2C” protocol (bProtocolNum=82h) have not been defined yet.

The response to this message is the RDR_to_PC_Parameters message.
This message could return any of the following errors.

<table>
<thead>
<tr>
<th>bmlICC Status</th>
<th>bmCommand Status</th>
<th>bError</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>bSlot does not exist</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>ICC_MUTE</td>
<td>No ICC present</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>BUSY_WITH_AUTO_SEQUENCE</td>
<td>Automatic sequence on-going</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>HW_ERROR</td>
<td>Hardware error</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Command Not Supported</td>
</tr>
<tr>
<td>0,1,2</td>
<td>1</td>
<td>CMD_SLOT_BUSY</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

4.1.8 PC_to_RDR_Escape

This command allows the CCID manufacturer to define and access extended features.
Information sent via this command is processed by the CCID control logic.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>6Bh</td>
<td>Size of abData field of this message</td>
</tr>
<tr>
<td>1</td>
<td>dwLength</td>
<td>4</td>
<td></td>
<td>Size of abData field of this message</td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td>00-FFh</td>
<td>Identifies the slot number for this command</td>
</tr>
<tr>
<td>6</td>
<td>bSeq</td>
<td>1</td>
<td>00-FFh</td>
<td>Sequence number for command.</td>
</tr>
<tr>
<td>7</td>
<td>abRFU</td>
<td>3</td>
<td></td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td>10</td>
<td>abData</td>
<td>Byte array</td>
<td>data block sent to the CCID</td>
<td></td>
</tr>
</tbody>
</table>

The response to this message is the RDR_to_PC_Escape message.
This message could return any of the following errors.

<table>
<thead>
<tr>
<th>bmlICC Status</th>
<th>bmCommand Status</th>
<th>bError</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 4.1.9 PC_to_RDR_IccClock

This command stops or restarts the clock.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>6Eh</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>dwLength</td>
<td>4</td>
<td>00000000h</td>
<td>Size of extra bytes of this message</td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td>00-FFh</td>
<td>Identifies the slot number for this command</td>
</tr>
<tr>
<td>6</td>
<td>bSeq</td>
<td>1</td>
<td>00-FFh</td>
<td>Sequence number for command.</td>
</tr>
<tr>
<td>7</td>
<td>bClockCommand</td>
<td>1</td>
<td></td>
<td>value =</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 00h restarts Clock</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 01h Stops Clock in the state shown in the bClockStop field of the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PC_to_RDR_SetParameters command and RDR_to_PC_Parameters message.</td>
</tr>
<tr>
<td>8</td>
<td>abRFU</td>
<td>2</td>
<td></td>
<td>Reserved for Future Use</td>
</tr>
</tbody>
</table>

The response to this message is the RDR_to_PC_StatusClock.

This message could return any of the following errors.

<table>
<thead>
<tr>
<th>bmICC Status</th>
<th>bmCommand Status</th>
<th>bError</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>bSlot does not exist</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>ICC_MUTE</td>
<td>No ICC present</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>BUSY_WITH_AUTO_SEQUENCE</td>
<td>Automatic sequence on-going</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>HW_ERROR</td>
<td>Hardware error</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Command Not Supported</td>
</tr>
<tr>
<td>0,1,2</td>
<td>1</td>
<td>CMD_SLOT_BUSY</td>
<td></td>
</tr>
</tbody>
</table>
4.1.10 PC_to_RDR_T0APDU

This command changes the parameters used to perform the transportation of APDU messages by the T=0 protocol. It effects the CLA (class) byte used when issuing a Get Response command or a Envelope command to the ICC.

This command is slot specific. It only effects the slot specified in the bSlot field. Slots, when not powered, will change back to using the default behaviour defined in the CCID class descriptor. Any newly inserted ICC will have the default behaviour until this command is issued for its slot.

Only CCIDs reporting a short or extended APDU level in the dwFeatures field of the CCID class descriptor may take this command into account.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>6Ah</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>dwLength</td>
<td>4</td>
<td>00000000h</td>
<td>Size of extra bytes of this message</td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td>00-FFh</td>
<td>Identifies the slot number for this command</td>
</tr>
<tr>
<td>6</td>
<td>bSeq</td>
<td>1</td>
<td>00-FFh</td>
<td>Sequence number for command.</td>
</tr>
<tr>
<td>7</td>
<td>bmChanges</td>
<td>1</td>
<td>00h,01h,02h,03h</td>
<td>The value is bitwise OR operation. Bit 0 is associated with field bClassGetResponse Bit 1 is associated with field bClassEnvelope Other bits are RFU. A bit cleared indicates that the associated field is not significant and that default behaviour defined in CCID class descriptor is selected. A bit risen indicates that the associated field is significant.</td>
</tr>
<tr>
<td>8</td>
<td>bClassGetResponse</td>
<td>1</td>
<td></td>
<td>Value to force the class byte of the header in a Get Response command. Value = FFh indicates that the class byte of the Get Response command echoes the class byte of the APDU.</td>
</tr>
<tr>
<td>9</td>
<td>bClassEnvelope</td>
<td>1</td>
<td></td>
<td>Value to force the class byte of the header in a Envelope command. Value = FFh indicates that the class byte of the Envelope command echoes the class byte of the APDU.</td>
</tr>
</tbody>
</table>
The response to this message is the RDR_to_PC_SlotStatus. This message could return any of the following errors.

<table>
<thead>
<tr>
<th>bmICC Status</th>
<th>bmCommand Status</th>
<th>bError</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>bSlot does not exist</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>ICC_MUTE</td>
<td>No ICC present</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>ICC_PROTOCOL_NOT_SUPPORTED</td>
<td>Protocol not managed</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>CMD_ABORTED</td>
<td>Command aborted by control pipe</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Command Not Supported</td>
</tr>
</tbody>
</table>

### 4.1.11 PC_to_RDR_Secure

This is a command message to allow entering the PIN for verification or modification.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>69h</td>
<td>Size of extra bytes of this message</td>
</tr>
<tr>
<td>1</td>
<td>dwLength</td>
<td>4</td>
<td></td>
<td>Identifies the slot number for this command</td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td>00-FFh</td>
<td>Sequence number for command.</td>
</tr>
<tr>
<td>6</td>
<td>bSeq</td>
<td>1</td>
<td>00-FFh</td>
<td>Used to extend the CCIDs Block Waiting Timeout for this current transfer. The CCID will timeout the block after “this number multiplied by the Block Waiting Time” has expired. This parameter is only used for character level exchanges.</td>
</tr>
<tr>
<td>8</td>
<td>wLevelParameter</td>
<td>2</td>
<td></td>
<td>Use changes depending on the exchange level reported by CCID in the functional descriptor:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• character level, size of expected data to be returned by the bulk-IN endpoint,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• TPDU level, RFU, = 0000h</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• short APDU level, RFU, = 0000h</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• extended APDU level, indicates if APDU begins or ends in this command:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000h the APDU command begins and ends with this command,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0001h the APDU command begins with this command, and continues in next PC_to_RDR_Secure,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0002h this abData field continues an APDU command and ends the APDU command,</td>
</tr>
</tbody>
</table>
### Offset Field Size Value Description

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| 10     | bPINOperation  | 1    | 00h, 04h  | Used to indicate the PIN operation:  
00h: PIN Verification  
01h: PIN Modification  
02h: Transfer PIN from secure CCID buffer  
In a character level exchange CCID, when the ICC responds to the ICC command header with the standard INS, the host will send this operation to tell the CCID to send the formatted PIN to the ICC.  
03h: Wait ICC response  
In a character level exchange CCID using protocol T=0, the ICC could respond to the ICC command header with 60h (indicating that the ICC is requesting an extended time) and that another status byte will be sent by the ICC. The host will send this operation to tell the CCID to wait for the response from the ICC. After getting the response, the “Transfer PIN from secure CCID buffer” operation would be used to complete the PIN sequence.  
04h: Cancel PIN function  
05h: Re-send last I-Block, valid only if protocol in use it T=1.  
06h: Send next part of APDU, valid only if protocol in use it T=1. |
| 11     | abPINDataStructure | Byte array | PIN Verification Data Structure or PIN Modification Data Structure                                                                                   |

If bPINOperation = 2,3, or 4, the abOperationDataStructure is empty. The abOperationDataStructure for 0 (PIN Verification), and 1 (PIN Modification) are described in §4.1.11.1 and §4.1.11.6.
### 4.1.11.1 PIN Verification Data Structure

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>bTimeOut</td>
<td>1</td>
<td></td>
<td>Number of seconds. If 00h then CCID default value is used.</td>
</tr>
<tr>
<td>12</td>
<td>bmFormatString</td>
<td>1</td>
<td></td>
<td>Several parameters for the PIN format options (defined in 4.1.11.3)</td>
</tr>
<tr>
<td>13</td>
<td>bmPINBlockString</td>
<td>1</td>
<td></td>
<td>Defines the length in bytes of the PIN block to present in the APDU command</td>
</tr>
<tr>
<td>14</td>
<td>bmPINLengthFormat</td>
<td>1</td>
<td></td>
<td>Allows the insertion of the PIN length in the APDU command (defined in 4.1.11.5)</td>
</tr>
<tr>
<td>15</td>
<td>wPINMaxExtraDigit</td>
<td>2</td>
<td>XXYYh</td>
<td>XX: Minimum PIN size in digit YY: Maximum PIN size in digit</td>
</tr>
<tr>
<td>17</td>
<td>bEntryValidationCondition</td>
<td>1</td>
<td></td>
<td>The value is a bit wise OR operation. 01h Max size reached 02h Validation key pressed 04h Timeout occurred</td>
</tr>
<tr>
<td>18</td>
<td>bNumberMessage</td>
<td>1</td>
<td>00h, 01h, FFh</td>
<td>Number of messages to display for the PIN Verification management. 00h no string 01h Message which index is indicated in bMsgIndex FFh default CCID message.</td>
</tr>
<tr>
<td>19</td>
<td>wLangId</td>
<td>2</td>
<td></td>
<td>Language used to display the messages. The 16 bit value is the same value used to select a language when getting a USB string descriptor.</td>
</tr>
<tr>
<td>21</td>
<td>bMsgIndex</td>
<td>1</td>
<td></td>
<td>Message index in the Reader CCID message table (should be 00h). The message is the prompt for the user to enter their PIN.</td>
</tr>
<tr>
<td>22</td>
<td>bTeoPrologue</td>
<td>3</td>
<td></td>
<td>T=1 I-block prologue field to use. Significant only if protocol in use is T=1.</td>
</tr>
<tr>
<td>25</td>
<td>abData</td>
<td>Byte array</td>
<td></td>
<td>APDU to send to the ICC</td>
</tr>
</tbody>
</table>

If the protocol being used is T=1 and the CCID is using character level or TPDU level exchanges, the CCID will behave in the following manner:

After PIN capture and APDU format with this PINCODE according to parameters given in the PIN Verification Data Structure:
a) If the APDU length is less than or equal to IFSC, the entire APDU becomes the INF field of an I-block,

b) If the APDU length is greater than IFSC, the first IFSC bytes of the APDU becomes the INF field of an I-block,

And the CCID computes the EDC for this I-block and sends the entire I-block, or computes the EDC while sending the prologue field (bTeoPrologue) and the INF field of the I-block then send the EDC.

In case b) the remainder of the APDU will be send by further PC_to_RDR_Secure commands with bPINOperation equals to 06h.

**4.1.11.2 Message table:**

For a CCID with display capabilities, the PIN verification and validation process usually involves the display of some message on the device to guide the user. ("ENTER PIN", etc..). The messages are accessible through an indexed table.

The way the Message table is initialized is out of the scope of this specification (escape command), but a compliant reader should reserve the first three message of its table for the following messages:

<table>
<thead>
<tr>
<th>Message Index</th>
<th>Message Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>PIN insertion prompt</td>
<td>&quot;ENTER SMARTCARD PIN&quot;</td>
</tr>
<tr>
<td>01</td>
<td>PIN Modification prompt</td>
<td>&quot;ENTER NEW PIN&quot;</td>
</tr>
<tr>
<td>02</td>
<td>NEW PIN Confirmation prompt</td>
<td>&quot;CONFIRM NEW PIN&quot;</td>
</tr>
</tbody>
</table>

The application has three options, for the message display:

- Forbid any message display by the CCID during PIN entry message
- Let the CCID displays default(s) message without control from the driver
- Specify the message index to be displayed. (1 message in the case of the PIN verification, 2 or 3 messages in the case of PIN Modification). This option allows use of specific message by the driver.

<table>
<thead>
<tr>
<th>bNumberMessage</th>
<th>bMsgIndex(es)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00h</td>
<td>N.A.</td>
<td>CCID does not display any message</td>
</tr>
<tr>
<td>01h</td>
<td>00h</td>
<td>CCID displays one message which index is defined by bMsgIndex (should be 00h, but is open to allow option like multiple language support, etc...) PIN Verification only</td>
</tr>
<tr>
<td>02h</td>
<td>00h, 01h, or 01h, 02h</td>
<td>CCID displays two messages which index are defined by bMsgIndex (should be 00h, and 01h but is open to allow option like multiple language support, etc..)</td>
</tr>
</tbody>
</table>
4. CCID Messages

4.11.3 bmFormatString description

This field provides several mechanisms in order to map the PIN code information. The information provides a system that allows different PIN representation (shift binary…). Moreover, in some cases, the device should have more parameters to complete this mapping (PIN justification and PIN format type for ASCII, Binary and BCD conversion).

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>The system units’ type indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If 0: the system units are bits</td>
</tr>
<tr>
<td></td>
<td>If 1: the system units are bytes</td>
</tr>
<tr>
<td></td>
<td>This bit quantifies the next parameter (unit moving).</td>
</tr>
</tbody>
</table>

| Bit 6 – 3 | Define the PIN position after format in the APDU command (relative to the first data after Lc). The position is based on the system units’ type indicator (maximum 1111 for fifteen system units). |

<table>
<thead>
<tr>
<th>Bit 2</th>
<th>Bit mask for the PIN justification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If 0: Left justify data</td>
</tr>
<tr>
<td></td>
<td>If 1: Right justify data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 1-0</th>
<th>Bit wise for the PIN format type:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>00: binary</td>
</tr>
<tr>
<td></td>
<td>01: BCD</td>
</tr>
<tr>
<td></td>
<td>10: ASCII</td>
</tr>
</tbody>
</table>

In the context of the PC_to_RDR_Secure command and selecting the PIN format type with the bmFormatString field, binary means that one digit is coded on one byte, while for BCD 2 digits would be coded on 1 byte.

Example: If the PIN presented is 12345

bmFormatString bit bit 1-0 = Binary --> the command to the ICC is going to look like

```
<Command header> <length> 01 02 03 04 05 ....
```

bmFormatString bit bit 1-0 = BCD --> the command to the ICC is going to look like

```
<Command header> <length> 12 34 5X .... (where X represents the padding character)
```
4.1.11.4 bmPINBlockString

This field provides the PIN block size and the PIN length size information.

| Bit 7 - 4 | Size in bits of the PIN length inserted in the APDU command. (If 0h, then the effective pin length is not inserted in the APDU command) |
|----------------------|
| Bit 3 - 0 | PIN length information: PIN block size in bytes after justification and formatting. |

4.1.11.5 bmPINLengthFormat

Some ICCs include the effective PIN length in their APDU command (EMV ICCs for example). The bmPINLengthFormat provides the PIN length position in the APDU command.

| Bit 7-5 | RFU |
|----------------------|
| Bit 4 | The system units’ type indicator |
| If 0: the system units are bits |
| If 1: the system units are bytes |
| Bit 3 - 0 | Indicate the PIN length position in the APDU command according to the previous parameters (maximum 1111 for fifteen system units). |

4.1.11.6 PIN Modification Data Structure

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>bTimeOut</td>
<td>1</td>
<td></td>
<td>Number of seconds. If 00h then CCID default value is used.</td>
</tr>
<tr>
<td>12</td>
<td>bmFormatString</td>
<td>1</td>
<td></td>
<td>Several parameters for the PIN format options (defined in 4.1.11.3)</td>
</tr>
<tr>
<td>13</td>
<td>bmPINBlockString</td>
<td>1</td>
<td></td>
<td>Define the length of the PIN to present in the APDU command</td>
</tr>
<tr>
<td>14</td>
<td>bmPINLengthFormat</td>
<td>1</td>
<td></td>
<td>Allows the length PIN insertion in the APDU command (defined in 4.1.11.5)</td>
</tr>
<tr>
<td>15</td>
<td>bInsertionOffsetOld</td>
<td>1</td>
<td></td>
<td>Insertion position offset in byte for the current PIN</td>
</tr>
<tr>
<td>16</td>
<td>bInsertionOffsetNew</td>
<td>1</td>
<td></td>
<td>Insertion position offset in byte for the new PIN</td>
</tr>
<tr>
<td>17</td>
<td>wPINMaxExtraDigit</td>
<td>2</td>
<td>XXYY</td>
<td>XX: Minimum PIN size in digit</td>
</tr>
</tbody>
</table>

3 Some ICC uses a shift bit mechanism to format the PIN. In this case the PIN position parameters in the bFormatString are not redundant with the bInsertionOffsetOld or bInsertionOffsetNew.
<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>bConfirmPIN</td>
<td>1</td>
<td>00h,01h</td>
<td>Indicates if a confirmation is requested before acceptance of a new PIN (meaning that the user has to enter this new PIN twice before it is accepted) Indicates if the current PIN must be entered and set in the same APDU field or not. b0: (0/1) If 0 = No confirmation requested If 1 = Confirmation requested B1: (0/1) If 0 = No current PIN entry requested. (In this case, the bInsertioOffsetOld value mustn’t be taken into account.) If 1 = Current PIN entry requested b2 – b7 : RFU</td>
</tr>
<tr>
<td>20</td>
<td>bEntryValidationCondition</td>
<td>1</td>
<td></td>
<td>The value is a bit wise OR operation. 01h Max size reached 02h Validation key pressed 04h Timeout occurred</td>
</tr>
<tr>
<td>21</td>
<td>bNumberMessage</td>
<td>1</td>
<td>00h,01h</td>
<td>02h,03h, or FFh Number of messages to display for the PIN modify command. 00h: no message. 01h: Message which index is indicated in bMsgIndex1 (case where bConfirmPIN = 00h) 02h Messages which index are indicated in bMsgIndex1, and bMsgIndex2 (case where bConfirmPIN = 01h or 02h) 03h Messages which index are indicated in bMsgIndex1, bMsgIndex2, and bMsgIndex3 (case where bConfirmPIN = 03h) FFh default CCID message(s)</td>
</tr>
<tr>
<td>22</td>
<td>wLangId</td>
<td>2</td>
<td></td>
<td>Language used to display the messages. The 16 bit value is the same value used to select a language when getting a USB string descriptor.</td>
</tr>
</tbody>
</table>
### Offset Field Size Value Description

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>bMsgIndex1</td>
<td>1</td>
<td></td>
<td>Message index in the Reader message table (should be 00h or 01h). The message is the prompt for the user to enter its new PIN if bConfirmPIN=00h or its current PIN.</td>
</tr>
<tr>
<td>25</td>
<td>bMsgIndex2</td>
<td>1</td>
<td></td>
<td>Message index in the Reader message table (should be 01h or 02h). The message is the prompt for the user to enter its NEW PIN value if bConfirm=02h or 03h, and to re-enter its new PIN if bConfirm=01h.</td>
</tr>
<tr>
<td>26</td>
<td>bMsgIndex3</td>
<td>1</td>
<td></td>
<td>Message index in the Reader message table (should be 02h). The message is the prompt for the user to re-enter its NEW PIN value for confirmation purposes (only present if bNumberMessage = 3).</td>
</tr>
<tr>
<td></td>
<td>bTeoPrologue</td>
<td>3</td>
<td></td>
<td>T=1 I-block prologue field to use. Significant only if protocol in use is T=1.</td>
</tr>
<tr>
<td></td>
<td>abPINA_pdu</td>
<td>Byte array</td>
<td>APDU to send to the ICC</td>
<td></td>
</tr>
</tbody>
</table>

If the protocol being used is T=1 and the CCID is using character level or TPDU level exchanges, the CCID will behave in the following manner:

After new pin capture (and confirmation if requested) and APDU format with the PIN(s) value(s) according to parameters given in the PIN Modification Data Structure:

- a) If the APDU length is less than or equal to IFSC, the entire APDU becomes the INF field of an I-block,
- b) If the APDU length is greater than IFSC, the first IFSC bytes of the APDU becomes the INF field of an I-block,

And the CCID computes the EDC for this I-block and sends the entire I-block, or computes the EDC while sending the prologue field (bTeoPrologue) and the INF field of the I-block then send the EDC.

In case b) the remainder of the APDU will be send by further PC_to_RDR_Secure commands with bPINOperation equals to 06h.
In order to provide an efficient security to avoid PIN violation, only the INS ICC command for PIN verification or PIN modification is valid for the PC_to_RDR_Secure message. The CCID should reject any INS ICC command which is not 20h (Verify), or 24h (PIN change/unblock).

The wLangId field supports displaying messages in multiple languages. The requester specifies the desired language by using a sixteen-bit language id (LANGID defined by the USB-IF. The list of currently defined USB LANGIDs can be found at http://www.usb.org/developers/docs.html. To determine what languages the CCID supports, a standard request of Get Descriptor for String Index 0 for all languages returns an array of the two-byte LANGID codes supported by the CCID.

### 4.1.11.7 Send Next Part of APDU Data Structure

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>bTeoPrologue</td>
<td>3</td>
<td></td>
<td>T=1 I-block prologue field to use. Significant only if protocol in use is T=1.</td>
</tr>
</tbody>
</table>

### 4.1.11.8 Remark on character or TPDU level CCID implement for T=1

For a CCID working at character level, or at TPDU level, when T=1 is the protocol in use, both of the following behaviors are valid. It is up to the CCID designer to choose which behavior their CCID will exhibit.

1. Accept PC_to_Reader_Secure requests with bPINOperation equal to 05h or 06h and do the required behavior. OR
2. Fail a PC_to_RDR_Secure when the APDU size exceeds the IFSC, indicating the offset of bTeoPrologue field and fail a PC_to_RDR_Secure with bPINOperation equal to 06h, indicating the offset of bPINOperation field.

The response to this message is the RDR_to_PC_DataBlock. This message could return any of the following errors.

<table>
<thead>
<tr>
<th>bmICC Status</th>
<th>bmCommand Status</th>
<th>bError</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>bSlot does not exist</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>ICC_MUTE</td>
<td>No ICC present</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>BUSY_WITH_AUTO_SEQUENCE</td>
<td>Automatic sequence on-going</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>HW_ERROR</td>
<td>Hardware error</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Command Not Supported</td>
</tr>
<tr>
<td>0,1,2</td>
<td>1</td>
<td>CMD_SLOT_BUSY</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>XFR_PARITY_ERROR</td>
<td>parity error on ATR</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>ICC_MUTE</td>
<td>ICC mute (Time out)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>CMD_ABORTED</td>
<td>Command aborted by control pipe</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Command Not Supported</td>
</tr>
</tbody>
</table>
4.1.12 PC_to_RDR_Mechanical

This command is used to manage motorized type CCID functionality. The Lock Card function is used to hold the ICC. This prevents an ICC from being easily removed from the CCID. The Unlock Card function is used to remove the hold initiated by the Lock Card function.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>71h</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>dwLength</td>
<td>4</td>
<td>00000000h</td>
<td>Size of extra bytes of this message</td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td>00-FFh</td>
<td>Identifies the slot number for this command</td>
</tr>
<tr>
<td>6</td>
<td>bSeq</td>
<td>1</td>
<td>00-FFh</td>
<td>Sequence number for command.</td>
</tr>
<tr>
<td>7</td>
<td>bFunction</td>
<td>1</td>
<td>04h-05h</td>
<td>This value corresponds to the mechanical function being requested</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01h – Accept Card</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>02h – Eject Card</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>03h – Capture Card</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>04h – Lock Card</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>05h – Unlock Card</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Values 01-03h are included for completeness but these functions are not</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>covered by this release of the specification</td>
</tr>
<tr>
<td>8</td>
<td>abRFU</td>
<td>2</td>
<td>00h</td>
<td>Reserved for Future Use</td>
</tr>
</tbody>
</table>

The response to this message is the RDR_to_PC_SlotStatus message.

This message could return any of the following errors.

<table>
<thead>
<tr>
<th>bmICC_Status</th>
<th>bmCommand_Status</th>
<th>bError</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>bSlot does not exist</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>ICC_MUTE</td>
<td>No ICC present</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>BUSY_WITH_AUTO_SEQUENCE</td>
<td>Automatic sequence on-going</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>HW_ERROR</td>
<td>Hardware error</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Command Not Supported</td>
</tr>
<tr>
<td>0,1,2</td>
<td>1</td>
<td>CMD_SLOT_BUSY</td>
<td></td>
</tr>
</tbody>
</table>
4.1.13 PC_to_RDR_Abort

This command is used with the Control pipe Abort request to tell the CCID to stop any current transfer at the specified slot and return to a state where the slot is ready to accept a new command pipe Bulk-OUT message. See discussion in section 3.7.1 Abort.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>72h</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>dwLength</td>
<td>4</td>
<td>0000000000</td>
<td>Size of extra bytes of this message</td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td>00-FFh</td>
<td>Identifies the slot number for this command</td>
</tr>
<tr>
<td>6</td>
<td>bSeq</td>
<td>1</td>
<td>00-FFh</td>
<td>Sequence number for command.</td>
</tr>
<tr>
<td>7</td>
<td>abRFU</td>
<td>3</td>
<td>00h</td>
<td>Reserved for Future Use</td>
</tr>
</tbody>
</table>

The response to this message is the RDR_to_PC_SlotStatus message.

This message could return any of the following errors.

<table>
<thead>
<tr>
<th>bmICC Status</th>
<th>bmCommand Status</th>
<th>bError</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>bSlot does not exist</td>
</tr>
<tr>
<td>0,1,2</td>
<td>1</td>
<td>0</td>
<td>Command Not Supported</td>
</tr>
</tbody>
</table>

4.1.14 PC_to_RDR_SetDataRateAndClockFrequency

This command is used to manually set the data rate and clock frequency of a specific slot.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>73h</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>dwLength</td>
<td>4</td>
<td>0000000000</td>
<td>Size of extra bytes of this message</td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td>00-FFh</td>
<td>Identifies the slot number for this command</td>
</tr>
<tr>
<td>6</td>
<td>bSeq</td>
<td>1</td>
<td>00-FFh</td>
<td>Sequence number for command.</td>
</tr>
<tr>
<td>7</td>
<td>abRFU</td>
<td>3</td>
<td>00h</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td>10</td>
<td>dwClockFrequency</td>
<td>4</td>
<td></td>
<td>ICC clock frequency in KHz encoded as a little endian value</td>
</tr>
<tr>
<td>14</td>
<td>dwDataRate</td>
<td>4</td>
<td></td>
<td>ICC data rate in bpd encoded as a little endian integer value</td>
</tr>
</tbody>
</table>

The response to this message is the RDR_to_PC_DataRateAndClockFrequency message.

This message could return any of the following errors.
### 4.2 CCID Bulk-IN Messages

#### 4.2.1 Status and Error Reporting in Bulk-IN messages

Each Bulk-IN Message contains both the value of the slot status register and the value of the slot error register.

**Slot Status register:**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bmICCStatus</td>
<td>2 bit</td>
<td>0, 1, 2</td>
<td>0 - An ICC is present and active (power is on and stable, RST is inactive)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 - An ICC is present and inactive (not activated or shut down by hardware error)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2 - No ICC is present</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3 - RFU</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>4 bits</td>
<td></td>
<td>RFU</td>
</tr>
<tr>
<td>6</td>
<td>bmCommandStatus</td>
<td>2 bits</td>
<td>0, 1, 2</td>
<td>0 - Processed without error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 - Failed (error code provided by the error register)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2 - Time Extension is requested</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3 - RFU</td>
</tr>
</tbody>
</table>

When the bmCommandStatus field is 0 indicating the command Processed without Error or when the bmCommand field is an RFU value, then the slot’s error register is RFU.

When the bmCommandStatus field is 2 indicating a Time extension is requested, then the slot’s error register contains the BWI value if the protocol is T=1 or the WI value if the protocol is T=0.

When the bmCommandStatus field is 1 indicating the command failed, then the slot’s error register is set with a signed 8-bit value.

A command fails when:
• The CCID does not support this command. Then the CCID sets the Slot Error register to ‘0’ (index of bMessageType field).
• The CCID cannot parse one parameter or the ICC is not supporting one parameter. Then the Slot Error register contains the index of the first bad parameter as a positive number (1-127). For instance, if the CCID receives an ICC command to an unimplemented slot, then the Slot Error register shall be set to ‘5’ (index of bSlot field).
• The CCID aborts the command before posting the Bulk-IN message header. Then the CCID sets the Slot Error register to CMD_ABORTED and dwLength field to ‘00000000h’.
• The command returns an error code as indicated by the command description (See section 4.1 “CCID Command Pipe Bulk-OUT Messages”).

The appendix I summarizes the Slot Error register values.

The CCID can abort a command after posting the Bulk-IN message header and before posting the complete message, by posting a short packet or a zero-length packet. This is called an “interrupted” partial response. The host driver shall interpret receiving a short packet or a zero-length packet ending an incomplete message as a failed command with CMD_ABORTED report. Then it shall stop sending Bulk-IN requests to complete this command.

4.2.2 Bulk-IN message details

<table>
<thead>
<tr>
<th>Message Name</th>
<th>bMessageType</th>
<th>Section</th>
<th>Command Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDR_to_PC_DataBlock</td>
<td>80h</td>
<td>4.2.2.1</td>
<td>PC_to_RDR_IccPowerOn</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PC_to_RDR_XfrBlock</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PC_to_RDR_Secure</td>
</tr>
<tr>
<td>RDR_to_PC_SlotStatus</td>
<td>81h</td>
<td>4.2.2.2</td>
<td>PC_to_RDR_IccPowerOff</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PC_to_RDR_GetSlotStatus</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PC_to_RDR_Mechanical</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PC_to_RDR_T0APDU</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PC_to_RDR_Abort and Class specific ABORT request</td>
</tr>
<tr>
<td>RDR_to_PC_Parameters</td>
<td>82h</td>
<td>4.2.2.3</td>
<td>PC_to_RDR_GetParameters</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PC_to_RDR_ResetParameters</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PC_to_RDR_SetParameters</td>
</tr>
<tr>
<td>RDR_to_PC_Escape</td>
<td>83h</td>
<td>4.2.2.4</td>
<td>PC_to_RDR_Escape</td>
</tr>
<tr>
<td>RDR_to_PC_DataRateAndClockFrequency</td>
<td>84h</td>
<td></td>
<td>PC_to_RDR_SetDataRateAndClockFrequency</td>
</tr>
</tbody>
</table>
4.2.2.1 RDR_to_PC_DataBlock

This message is sent by the device in response to several different command messages.

This message is sent in response to “PC_to_RDR_IccPowerOn.” This response is the answer to reset (ATR) data associated with the ICC power on.

This message is sent in response to “PC_to_RDR_Secure” or “PC_to_RDR_XfrBlock.” This response represents the ICC response to the ICC command. The response is in the following format: The response data will contain the optional data returned by the ICC, followed by the 2 byte-size status words SW1-SW2.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>80h</td>
<td>Indicates that a data block is being sent from the CCID</td>
</tr>
<tr>
<td>1</td>
<td>dwLength</td>
<td>4</td>
<td></td>
<td>Size of extra bytes of this message</td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td></td>
<td>Identifies the slot number for this command</td>
</tr>
<tr>
<td>6</td>
<td>bSeq</td>
<td>1</td>
<td></td>
<td>Sequence number for the corresponding command.</td>
</tr>
<tr>
<td>7</td>
<td>bStatus</td>
<td>1</td>
<td></td>
<td>Slot status register as defined in section 4.2.1</td>
</tr>
<tr>
<td>8</td>
<td>bError</td>
<td>1</td>
<td></td>
<td>Slot error register as defined in section 4.2.1</td>
</tr>
<tr>
<td>9</td>
<td>bChainParameter</td>
<td>1</td>
<td></td>
<td>Use changes depending on the exchange level reported by the CCID in the functional descriptor:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• character level, TPDU level, short APDU level, this field is RFU and =00h.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• extended APDU level, indicates if the response is complete, to be continued or if the command APDU can continue 00h the response APDU begins and ends in this command 01h the response APDU begins</td>
</tr>
</tbody>
</table>

4. CCID Messages
4.2.2.2 RDR_to_PC_SlotStatus:

This message is sent by the device in response to a “PC_to_RDR_GetSlotStatus”, a “PC_to_RDR_IccPowerOff”, a “PC_to_RDR_Mechanical”, a “PC_to_RDR_IccClock”, a “PC_to_RDR_T0APDU” message, or when the CCID has completed aborting a slot after receiving both the Class Specific ABORT request and the PC_to_RDR_Abort message.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>81h</td>
<td>bMessageType 1  81h</td>
</tr>
<tr>
<td>1</td>
<td>dwLength</td>
<td>4</td>
<td>00000000000h</td>
<td>Size of extra bytes of this message</td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td>Same as Bulk-OUT message</td>
<td>Identifies the slot number for this command</td>
</tr>
<tr>
<td>6</td>
<td>bSeq</td>
<td>1</td>
<td>Same as Bulk-OUT message</td>
<td>Sequence number for the corresponding command</td>
</tr>
</tbody>
</table>
4.2.2.3 RDR_to_PC_Parameters

This message is sent by the device in response to the messages (1) “PC_to_RDR_GetParameters”, (2)“PC_to_RDR_ResetParameters”, and (3) “PC_to_RDR_SetParameters”.

The descriptor includes the slot configuration defaults. These can be changed with the Set Parameters message and reset to the original states with the Reset Parameters message.

If the CCID does not support the “PC_to_RDR_ResetParameters”, or “PC_to_RDR_SetParameters” command, the “Command Not Supported” status shall be returned in the bStatus and bError fields (bStatus=01xx0000, bError=00h) will be set and no bProtocolNum or bProtocolDataStructure fields will be returned (dwLength=00000000h).

If the CCID supports the “PC_to_RDR_SetParameters” command, and the command requests a change to an unchangeable parameter, or requests to change a changeable parameter to an invalid value, the CCID will change no parameters and will return the RDR_to_PC_Parameters message with the message offset of the first parameter in error in the bError field. The RDR_to_PC_Parameters message will report the current settings.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>82h</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>dwLength</td>
<td>4</td>
<td></td>
<td>Size of extra bytes of this message</td>
</tr>
</tbody>
</table>
### Offsets

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td>Same as Bulk-OUT message</td>
<td>Identifies the slot number for this command</td>
</tr>
<tr>
<td>6</td>
<td>bSeq</td>
<td>1</td>
<td>Same as Bulk-OUT message</td>
<td>Sequence number for the corresponding command.</td>
</tr>
<tr>
<td>7</td>
<td>bStatus</td>
<td>1</td>
<td></td>
<td>Slot status register as defined in section [4.2.1]</td>
</tr>
<tr>
<td>8</td>
<td>bError</td>
<td>1</td>
<td></td>
<td>Slot error register as defined in section [4.2.1]</td>
</tr>
</tbody>
</table>
| 9      | bProtocolNum| 1    | 00h, 01h | Specifies what protocol data structure follows.  
00h = Structure for protocol T=0  
01h = Structure for protocol T=1  
The following values are reserved for future use.  
80h = Structure for 2-wire protocol  
81h = Structure for 3-wire protocol  
82h = Structure for I2C protocol |
| 10     | abProtocolDataStructure | Byte Array | Protocol Data Structure |

### Protocol Data Structure for Protocol T=0 (bProtocolNum=0)(dwLength=00000005h)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| 10     | bmFindexDindex | 1    |         | B7-4 – FI – Index into the table 7 in ISO/IEC 7816-3:1997 selecting a clock rate conversion factor  
B3-0 – DI - Index into the table 8 in ISO/IEC 7816-3:1997 selecting a baud rate conversion factor |
| 11     | bmTCCKST0     | 1    | 00h, 02h | For T=0 , B0 – 0b, B7-2 – 000000b  
B1 – Convention used (b1=0 for direct, b1=1 for inverse) |
### 4. CCID Messages

#### USB Chip/Smart Card Interface Devices, Revision 1.00

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>bGuardTimeT0</td>
<td>1</td>
<td>00-FFh</td>
<td>Extra Guardtime between two characters. Add 0 to 254 etu to the normal guardtime of 12etu. FFh is the same as 00h.</td>
</tr>
<tr>
<td>13</td>
<td>bWaitingIntegerT0</td>
<td>1</td>
<td>00-FFh</td>
<td>WI for T=0 used to define WWT</td>
</tr>
</tbody>
</table>
| 14     | bClockStop             | 1    | 00-03h    | ICC Clock Stop Support  
00 = Stopping the Clock is not allowed  
01 = Stop with Clock signal Low  
02 = Stop with Clock signal High  
03 = Stop with Clock either High or Low |

#### Protocol Data Structure for Protocol T=1 (bProtocolNum=1)(dwLength=00000007h)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| 10     | bmFindexDindex         | 1    |           | B7-4 – FI – Index into the table 7 in ISO/IEC 7816-3:1997 selecting a clock rate conversion factor  
B3-0 – DI  - Index into the table 8 in ISO/IEC 7816-3:1997 selecting a baud rate conversion factor |
| 11     | bmTCCKST1              | 1    | 10h, 11h, 12h, 13h | For T=1, B7-2 – 000100b  
B0 – Checksum type (b0=0 for LRC, b0=1 for CRC  
B1 – Convention used (b1=0 for direct, b1=1 for inverse) |
| 12     | bGuardTimeT1           | 1    | 00-FFh    | Extra Guardtime (0 to 254 etu between two characters). If value is FFh, then guardtime is reduced by 1.                                      |
| 13     | bmWaitingIntegers T1   | 1    | 00-9Fh    | B7-4 = BWI  
B3-0 = CWI                                                                                                                                     |
| 14     | bClockStop             | 1    | 00-03h    | ICC Clock Stop Support  
00 = Stopping the Clock is not allowed  
01 = Stop with Clock signal Low  
02 = Stop with Clock signal High  
03 = Stop with Clock either High or Low |
<p>| 15     | bIFSC                  | 1    | 00-FEh    | Size of negotiated IFSC                                                                                                                      |</p>
<table>
<thead>
<tr>
<th></th>
<th>bNadValue</th>
<th></th>
<th>00-FFh</th>
<th>Nad value used by CCID</th>
</tr>
</thead>
</table>

Protocol Data Structures for “2-wire” protocol (bProtocolNum=80h), “3-wire” protocol (bProtocolNum=81h), and “I2C” protocol (bProtocolNum=82h) have not been defined yet.
4.2.2.4 RDR_to_PC_Escape

This message is sent by the device in response to the message “PC_to_RDR_Escape”.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>83h</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>dwLength</td>
<td>4</td>
<td></td>
<td>Size of extra bytes of this message</td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td></td>
<td>Same as Bulk-OUT message</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Identifies the slot number for this command</td>
</tr>
<tr>
<td>6</td>
<td>bSeq</td>
<td>1</td>
<td></td>
<td>Same as Bulk-OUT message</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sequence number for the corresponding command.</td>
</tr>
<tr>
<td>7</td>
<td>bStatus</td>
<td>1</td>
<td></td>
<td>Slot status register as defined in section 4.2.1</td>
</tr>
<tr>
<td>8</td>
<td>bError</td>
<td>1</td>
<td></td>
<td>Slot error register as defined in section 4.2.1</td>
</tr>
<tr>
<td>9</td>
<td>bRFU</td>
<td>1</td>
<td>00h</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td>10</td>
<td>abData</td>
<td></td>
<td></td>
<td>Data sent from CCID</td>
</tr>
</tbody>
</table>

4.2.2.5 RDR_to_PC_DataRateAndClockFrequency

This message is sent by the device in response to the message “PC_to_RDR_SetDataRateAndClockFrequency”.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>84h</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>dwLength</td>
<td>4</td>
<td>00000008h</td>
<td>Size of extra bytes of this message</td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td></td>
<td>Same as Bulk-OUT message</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Identifies the slot number for this command</td>
</tr>
<tr>
<td>6</td>
<td>bSeq</td>
<td>1</td>
<td></td>
<td>Same as Bulk-OUT message</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sequence number for the corresponding command.</td>
</tr>
<tr>
<td>7</td>
<td>bStatus</td>
<td>1</td>
<td></td>
<td>Slot status register as defined in section 4.2.1</td>
</tr>
<tr>
<td>8</td>
<td>bError</td>
<td>1</td>
<td></td>
<td>Slot error register as defined in section 4.2.1</td>
</tr>
<tr>
<td>9</td>
<td>bRFU</td>
<td>1</td>
<td>00h</td>
<td>Reserved for Future Use</td>
</tr>
</tbody>
</table>
4.3 CCID Interrupt-IN Messages

The Interrupt-IN endpoint is used to notify the host of events that may occur asynchronously and outside the context of a command-response exchange between host and CCID. If the host has sent a Bulk-Out message and is waiting for a Bulk-IN message in response, and one of these events occurs, then the Bulk-IN message may have duplicate information related to the event.

Table 4-3. Summary of Interrupt-In Messages

<table>
<thead>
<tr>
<th>Message Name</th>
<th>bMessageType</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDR_to_PC_NotifySlotChange</td>
<td>50h</td>
<td>4.3.1</td>
</tr>
<tr>
<td>RDR_to_PC_HardwareError</td>
<td>51h</td>
<td>4.3.2</td>
</tr>
</tbody>
</table>

4.3.1 RDR_to_PC_NotifySlotChange

This message is sent whenever the CCID device detects a change in the insertion status of an ICC slot. If an ICC is either inserted or removed from a slot, this message must be sent. The presence of this message means to the host driver that a change has occurred. It is possible for more than one change to occur between delivery of RDR_to_PC_NotifySlotChange messages.

When 1) a configuration other than ‘0’ is selected or 2) when the USB bus is resumed from a suspended state, both the CCID and the host driver must make identical assumptions about the state of the ICC slots. For simplicity, this specification requires that both CCID and host driver shall presume that all slots are empty. Therefore, after either a configuration change or resumption from suspend, the CCID shall report all occupied ICC slots using this message. See discussion in APPENDIX G Application Notes, 10.1 Suspend Behavior.
### 4.3.2 RDR_to_PC_HardwareError

This message is sent when any bit in the bHardwareErrorCode field is set. If this message is sent when there is no “outstanding” command, the bSeq field will be undefined.

When a CCID detects an over current condition, it will do the following. The CCID will deactivate the slot with the over current. If the CCID supports this message, it will send this message. If a Bulk-OUT command is active on the slot that has the over current, then that command is stopped and the inactive state of the slot and an over current error is returned in the bStatus and bError fields of the Bulk-IN response for that command.

If the CCID does not support this message and an error condition occurs, then the condition will be reported on the next Bulk-Out command.

---

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
Offset | Field               | Size | Value  | Description                                                                 |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>51h</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>bSlot</td>
<td>1</td>
<td>00h-FFh</td>
<td>ICC slot number</td>
</tr>
<tr>
<td>2</td>
<td>bSeq</td>
<td>1</td>
<td>00h-FFh</td>
<td>Sequence number of bulk out command when the hardware error occurred</td>
</tr>
<tr>
<td>3</td>
<td>bHardwareErrorCode</td>
<td>1</td>
<td>??h</td>
<td>The value is a bitwise OR operation performed on the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 01h Overcurrent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
</tr>
</tbody>
</table>
5. APPENDIX A Sequences common to all levels of message exchange

Note: The sequences in Appendices A-D are not exhaustive of all the possible cases. They are only meant as examples of several of the possible sequences the CCID must be able to handle.

Note: The fields shown in the messages in the sequences in Appendices A-D are only those fields applicable to that message.
Insert and removal sequence if interrupt IN endpoint supported by the CCID

**USB HOST**

- **Interrupt - IN**
  - RDR_to_PC_NotifySlotChange
  - bmSlotICCState = xxxxxx11b

**CHIP CARD INTERFACE DEVICE**

- **Insert ICC in Slot#0**
- **Bulk - IN**
  - RDR_to_PC_DataBlock
  - bSlot=00h, bSeq=08h
  - bStatus=00h=ICC present and active
  - abData=answer to reset (ATR) data
  - dwLength=size in bytes of ATR data

**Bulk - OUT**

- PC_to_RDR_DataBlock
  - bSlot=00h, bSeq=09h
  - abData=card command
  - dwLength=size in bytes of ICC command

**Remove ICC from Slot#0**

- **Interrupt Message**
  - RDR_to_PC_NotifySlotChange
  - bmSlotICCState = xxxxxx10b
Normal command sequence for a CCID with queuing capabilities

(CCID allows a maximum of 3 active commands)

Note: Because the CCID reported bMaxCCIDBusySlots equals 3 – at any time, only 3 active commands are allowable. Responses for different slot commands can be send by the CCID asynchronously upon command completion. The Host keeps information about the active commands and reset it after getting a response. If status byte of a response has Time Extension bit set, the Host will not decrease number of active commands and just wait for a new response with the same bSeq number.
6. APPENDIX B “Character Level” message exchange sequences

Note: The sequences in Appendices A-D are not exhaustive of all the possible cases. They are only meant as examples of several of the possible sequences the CCID must be able to handle.

Note: The fields shown in the messages in the sequences in Appendices A-D are only those fields applicable to that message.

Note: These sequences show exchanges with an ICC in slot bSlot=01h. This is the “second” slot of a multiple slot CCID. Exchanges with an ICC in the “first” slot of a multiple slot CCID or in the only slot of a single slot CCID would require using bSlot=00h.
Protocol T=0 sequence case 2 command (expect data from the ICC) with the length not accepted by the ICC.

**USB HOST**

**CHIP CARD INTERFACE DEVICE**

<table>
<thead>
<tr>
<th>USB HOST</th>
<th>CHIP CARD INTERFACE DEVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bulk-OUT</strong></td>
<td><strong>Bulk - IN</strong></td>
</tr>
<tr>
<td><strong>PC_to_RDR_XfrBlock</strong></td>
<td><strong>RDR_to_PC_DataBlock</strong></td>
</tr>
<tr>
<td>bSlot = 01h, bSeq = 0Ah</td>
<td>bSlot = 01h, bSeq = 0Ah</td>
</tr>
<tr>
<td>wLevelParameter = 0002h</td>
<td>bStatus = 00h = ICC present and active</td>
</tr>
<tr>
<td>bBW1 byte = 00h</td>
<td>dwLength = 00000005h</td>
</tr>
<tr>
<td>dwLength = 00000005h</td>
<td>abData = (CLA,INS,P1,P2,P3) = The ICC command with P3 the wrong size.</td>
</tr>
<tr>
<td>abData = (CLA,INS,P1,P2,P3) = The ICC command with the length received from the card.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>USB HOST</th>
<th>CHIP CARD INTERFACE DEVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bulk-OUT</strong></td>
<td><strong>Bulk - IN</strong></td>
</tr>
<tr>
<td><strong>PC_to_RDR_XfrBlock</strong></td>
<td><strong>RDR_to_PC_DataBlock</strong></td>
</tr>
<tr>
<td>bSlot = 01h, bSeq = 0Bh</td>
<td>bSlot = 01h, bSeq = 0Bh</td>
</tr>
<tr>
<td>wLevelParameter = 0001h</td>
<td>bStatus = 00h = ICC present and active</td>
</tr>
<tr>
<td>bBW1 byte = 00h</td>
<td>dwLength = 00000001h</td>
</tr>
<tr>
<td>dwLength = 00000005h</td>
<td>abData = (CLA,INS,P1,P2,XX) = The ICC command with the length received from the card.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>USB HOST</th>
<th>CHIP CARD INTERFACE DEVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bulk-OUT</strong></td>
<td><strong>Bulk - IN</strong></td>
</tr>
<tr>
<td><strong>PC_to_RDR_XfrBlock</strong></td>
<td><strong>RDR_to_PC_DataBlock</strong></td>
</tr>
<tr>
<td>bSlot = 01h, bSeq = 0Ch</td>
<td>bSlot = 01h, bSeq = 0Ch</td>
</tr>
<tr>
<td>wLevelParameter = XX+2</td>
<td>bStatus = 00h = ICC present and active</td>
</tr>
<tr>
<td>bBW1 byte = 00h</td>
<td>dwLength = XX+2</td>
</tr>
<tr>
<td>dwLength = 00000000h</td>
<td>abData = (XX bytes of Data,90h,00h) = ICC’s data and status response.</td>
</tr>
</tbody>
</table>

6 APPENDIX B “Character Level” message exchange sequences
Protocol T=0 sequence case 2 command (expect data from the ICC),

Procedure byte = not INS

---

**USB HOST**

**CHIP CARD INTERFACE DEVICE**

**PC_to_RDR_XfrBlock**
- bSlot = 01h, bSeq = 0Ch
- wLevelParameter = 0001h
- bBWl = 00h
- dwLength = 00000005h
- abData = (CLA,INS,P1,P2,P3) = ICC command

**PC_to_RDR_XfrBlock**
- bSlot = 01h, bSeq = 0Dh
- wLevelParameter = 0001h
- bBWl = 00h
- dwLength = 00000000h
- abData is absent

**PC_to_RDR_XfrBlock**
- bSlot = 01h, bSeq = 0Dh
- wLevelParameter = 0001h
- bBWl = 00h
- dwLength = 00000000h
- abData is absent

**PC_to_RDR_XfrBlock**
- bSlot = 01h, bSeq = 2Dh
- wLevelParameter = 0001h
- bBWl = 00h
- dwLength = 00000000h
- abData is absent

**RDR_to_PC_DataBlock**
- bSlot = 01h, bSequence = 0Ch
- bStatus = 00h = ICC present and active
- dwLength = 00000001h
- abData field = (Not INS) = procedure byte.

**RDR_to_PC_DataBlock**
- bSlot = 01h, bSeq = 0Dh
- bStatus = 00h = ICC present and active
- dwLength = 00000001h
- abData = (Data) = one byte of ICC data.

**RDR_to_PC_DataBlock**
- bSlot = 01h, bSeq = 2Dh
- bStatus = 00h = ICC present and active
- dwLength = 00000001h
- abData = (00h) = SW2 status byte

---

6 APPENDIX B “Character Level” message exchange sequences
Protocol T=0 sequence case 3 command (send data to the ICC),
Procedure byte = not INS

- **USB HOST**
- **CHIP CARD INTERFACE DEVICE**

**Bulk-OUT**

PC_to_RDR_XfrBlock
bSlot = 01h, bSeq = 0Ch
wLevelParameter = 0001h
bBWlbyte = 00h
dwLength = 00000005h
abData = (CLA,INS,P1,P2,P3) = ICC command

Bulk-OUT

PC_to_RDR_XfrBlock
bSlot = 01h, bSeq = 0Dh
wLevelParameter = 0001h
bBWlbyte = 00h
dwLength = 00000001h
abData = (Data) = one byte of ICC data

**Bulk-OUT** xx

PC_to_RDR_XfrBlock
bSlot # 01h, bSeq = 0Eh
wLevelParameter = 0001h
bBWlbyte = 00h
dwLength = 00000001h
abData = (Data) = one byte of ICC data

**Bulk-OUT**

PC_to_RDR_XfrBlock
bSlot = 01h, bSeq = 2Dh
wLevelParameter = 0001h
bBWlbyte = 00h
dwLength = 00000000h
abData is absent

**Bulk-IN**

RDR_to_PC_DataBlock
bSlot = 01h, bSeq = 0Ch
bStatus = 00h = ICC present and active
dwLength = 00000001h
abData field = (Not INS) = procedure byte.

**Bulk-OUT**

PC_to_RDR_XfrBlock
bSlot = 01h, bSeq = 0Dh
bStatus = 00h = ICC present and active
dwLength = 00000001h
abData field = (Not INS) = procedure byte

**Bulk-OUT**

PC_to_RDR_XfrBlock
bSlot = 01h, bSeq = 2Dh
bStatus = 00h = ICC present and active
dwLength = 00000000h
abData is absent

**Bulk-IN**

RDR_to_PC_DataBlock
bSlot = 01h, bSeq = 0Ch
bStatus = 00h = ICC present and active
dwLength = 00000001h
abData field = (Not INS) = Not INS for all transfers except the last one where 90h is returned.
Protocol T=0 sequence case 3 command (send data to the ICC),

Procedure byte = INS

**USB HOST**

Bulk-OUT

**PC_to_RDR_XfrBlock**

* bSlot = 01h, bSequence = 0Ch
* wLevelParameter = 0001h
* bBW1 byte=00h
* dwLength = 00000005h
  
  abData = (CLA,INS,P1,P2,P3) = ICC command

**CHIP CARD INTERFACE DEVICE**

Bulk - IN

**RDR_to_PC_DataBlock**

* bSlot = 01h, bSeq =0Ch
* bStatus=00h=ICC present and active
* dwLength = 00000001h
  
  abData field = (INS) = procedure byte.

**USB HOST**

Bulk-OUT

**PC_to_RDR_XfrBlock**

* bSlot = 01h, bSeq = 0Dh
* wLevelParameter = 0002h
* bBW1 byte=00h
* dwLength = P3
  
  abData = (Data(P3)) = P3 bytes of data

**CHIP CARD INTERFACE DEVICE**

Bulk - IN

**RDR_to_PC_DataBlock**

* bSlot = 01h, bSeq = 0Dh
* bStatus=00h=ICC present and active
* dwLength = 00000002h
  
  abData = (90h,00h) = ICC’s SW1 and SW2 response to command.
Protocol T=0 sequence case 4 command

(send data to the ICC, receive data from the ICC),

Procedure byte = INS

<table>
<thead>
<tr>
<th>USB HOST</th>
<th>CHIP CARD INTERFACE DEVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PC_to_RDR_XfrBlock</strong></td>
<td><strong>RDR_to_PC_DataBlock</strong></td>
</tr>
<tr>
<td>bSlot = 01h, bSeq = 10h</td>
<td>bSlot = 01h, bSeq = 10h</td>
</tr>
<tr>
<td>wLevelParameter = 0001h</td>
<td>bStatus=00h=ICC present and active</td>
</tr>
<tr>
<td>bBW1 byte=00h</td>
<td>dwLength = 00000001h</td>
</tr>
<tr>
<td>dwLength = 00000005h</td>
<td>abData field = (INS) = procedure byte</td>
</tr>
<tr>
<td>abData = (CLA,INS,P1,P2,P3) = ICC command</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bulk-OUT</th>
<th>Bulk-OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PC_to_RDR_XfrBlock</strong></td>
<td><strong>RDR_to_PC_DataBlock</strong></td>
</tr>
<tr>
<td>bSlot = 01h, bSeq = 11h</td>
<td>bSlot = 01h, bSeq = 11h</td>
</tr>
<tr>
<td>wLevelParameter = 0001h</td>
<td>bStatus=00h=ICC present and active</td>
</tr>
<tr>
<td>bBW1 byte=00h</td>
<td>dwLength = 00000001h</td>
</tr>
<tr>
<td>dwLength = P3+1</td>
<td>abData field = (61h) = SW1 status byte</td>
</tr>
<tr>
<td>abData = (Data(P3),Le) = P3 bytes of ICC data and 1 byte of Length of expected data</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bulk-OUT</th>
<th>Bulk-OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PC_to_RDR_XfrBlock</strong></td>
<td><strong>RDR_to_PC_DataBlock</strong></td>
</tr>
<tr>
<td>bSlot = 01h, bSeq = 12h</td>
<td>bSlot = 01h, bSeq = 12h</td>
</tr>
<tr>
<td>wLevelParameter = 0001h</td>
<td>bStatus=00h=ICC present and active</td>
</tr>
<tr>
<td>bBW1 byte=00h</td>
<td>dwLength = 00000001h</td>
</tr>
<tr>
<td>dwLength = 00000000h</td>
<td>abData field = (Le) = SW2 status bytes = length of data</td>
</tr>
<tr>
<td>abData = absent</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bulk-OUT</th>
<th>Bulk-OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PC_to_RDR_XfrBlock</strong></td>
<td><strong>RDR_to_PC_DataBlock</strong></td>
</tr>
<tr>
<td>bSlot = 01h, bSeq = 13h</td>
<td>bSlot = 01h, bSeq = 13h</td>
</tr>
<tr>
<td>wLevelParameter = 0001h</td>
<td>bStatus=00h=ICC present and active</td>
</tr>
<tr>
<td>bBW1 byte=00h</td>
<td>dwLength = 00000001h</td>
</tr>
<tr>
<td>dwLength = 00000005h</td>
<td>abData field = (C0h) = procedure byte</td>
</tr>
<tr>
<td>abData = (CLA,C0h,00h,00h,Le) = ICC “get response” command</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bulk-OUT</th>
<th>Bulk-OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PC_to_RDR_XfrBlock</strong></td>
<td><strong>RDR_to_PC_DataBlock</strong></td>
</tr>
<tr>
<td>bSlot = 01h, bSeq = 14h</td>
<td>bSlot = 01h, bSeq = 14h</td>
</tr>
<tr>
<td>LevelParameter = Le + 2</td>
<td>bStatus=00h=ICC present and active</td>
</tr>
<tr>
<td>bBW1 byte=00h</td>
<td>dwLength = Le + 2</td>
</tr>
<tr>
<td>dwLength = 00000000h</td>
<td>abData field = (Data(Le),90h,00h) = Le bytes of expected data and ICC’s SW1 and SW2 status bytes</td>
</tr>
<tr>
<td>abData is absent</td>
<td></td>
</tr>
</tbody>
</table>
Protocol T=1 sequence
(send I-block, receive I-block acknowledge),

**USB HOST**

**PC_to_RDR_XfrBlock**
- bSlot = 01h, bSeq = 40h
- wLevelParameter = 0003h
- bBW1 byte = 00h
- wLength = length of abData field
- abData = prologue field, Iblock data, and epilogue field.

**CHIP CARD INTERFACE DEVICE**

**Bulk-OUT**

**PC_to_RDR_XfrBlock**
- bSlot = 01h, bSeq = 41h
- wLevelParameter = LEN+1, or LEN+2 (depending on the size of the epilogue field)
- bBW1 byte = 00h
- dwLength = 00000000h
- abData is absent

**Bulk-IN**

**RDR_to_PC_DataBlock**
- bSlot = 01h, bSeq = 40h
- bStatus = 00h = ICC present and active
- dwLength = 0000003h
- abData = (NAD, PCB, LEN) = prologue field.

**Bulk-OUT**

**PC_to_RDR_XfrBlock**
- bSlot = 01h, bSeq = 41h
- wLevelParameter = LEN+1, or LEN+2 (depending on the size of the epilogue field)
- bBW1 byte = 00h
- dwLength = 00000000h
- abData is absent

**Bulk-IN**

**RDR_to_PC_DataBlock**
- bSlot = 01h, bSeq = 41h
- bStatus = 00h = ICC present and active
- dwLength = length of abData field
- abData = information field and epilogue field.
**APPENDIX B  "Character Level" message exchange sequences**

**Protocol T=1 sequence : Chaining**

---

**USB HOST**

**Bulk-OUT**

**PC_to_RDR_XfrBlock**
- `bSlot = 01h`
- `bSeq = 42h`
- `wLevelParameter = 0003h`
- `bBWl byte=00h`
- `dwLength = length of abData field`
- `abData = prologue field (0,1), l-block data(first), and epilogue field.`

**Bulk-OUT**

**PC_to_RDR_XfrBlock**
- `bSlot = 01h`
- `bSeq = 43h`
- `wLevelParameter = LEN+1, or LEN+2`
- (depending on the size of the epilogue field)
- `bBWl byte=00h`
- `dwLength = 00000000h`
- `abData is absent`

**Bulk-OUT**

**PC_to_RDR_XfrBlock**
- `bSlot = 01h`
- `bSeq = 44h`
- `wLevelParameter = 0003h`
- `bBWl byte=00h`
- `dwLength = length of abData field`
- `abData = prologue field (1,0), l-block data(second,last), and epilogue field.`

**Bulk-OUT**

**PC_to_RDR_XfrBlock**
- `bSlot = 01h`
- `bSeq = 45h`
- `wLevelParameter = LEN+1, or LEN+2`
- (depending on the size of the epilogue field)
- `bBWl byte=00h`
- `dwLength = 00000000h`
- `abData is absent`

---

**CHIP CARD INTERFACE DEVICE**

**Bulk-IN**

**RDR_to_PC_DataBlock**
- `bSlot = 01h`
- `bSeq = 42h`
- `bStatus=00h=ICC present and active`
- `dwLength = 00000003h`
- `abData = (NAD,PCB,LEN=00h)`
  - prologue field R(1).

**Bulk-IN**

**RDR_to_PC_DataBlock**
- `bSlot = 01h`
- `bSeq = 43h`
- `bStatus=00h=ICC present and active`
- `dwLength = length of abData field`
- `abData = Information field (empty)`
  - and Epilogue field of R-block

**Bulk-IN**

**RDR_to_PC_DataBlock**
- `bSlot = 01h`
- `bSeq = 44h`
- `bStatus=00h=ICC present and active`
- `dwLength = length of abData field`
- `abData = Information field (empty)`
  - and Epilogue field.

**Bulk-IN**

**RDR_to_PC_DataBlock**
- `bSlot = 01h`
- `bSeq = 45h`
- `bStatus=00h=ICC present and active`
- `dwLength = length of abData field`
- `abData = Information field (empty)`
  - and Epilogue field.
Protocol T=1 sequence: Waiting Time Extension

USB HOST

PC_to_RDR_XfrBlock
bSlot = 01h, bSeq = 46h
wLevelParameter = 0003h
bBWI byte = 00h
dwLength = length of abData field
abData = prologue field, I(0,0), lblock data, and epilogue field.

Bulk-OUT

PC_to_RDR_XfrBlock
bSlot = 01h, bSeq = 47h
wLevelParameter = LEN+1, or LEN+2 (depending on the size of the epilogue field) bBWI byte = 00h
dwLength = 00000000h
abData is absent

Bulk-OUT

PC_to_RDR_XfrBlock
bSlot = 01h, bSeq = 48h
wLevelParameter = 0003h
bBWI byte = ZZ
dwLength = length of abData field
abData = prologue field S(WTX response), Information field (1 byte), and epilogue field

Bulk-OUT

PC_to_RDR_XfrBlock
bSlot = 01h, bSeq = 49h
wLevelParameter = LEN+1, or LEN+2 (depending on the size of the epilogue field)
bBWI byte = 00h
dwLength = 00000000h
abData is absent

Bulk-OUT

Chip Card Interface Device

RDR_to_PC_DataBlock
bSlot = 01h, bSeq = 46h
bStatus=00h=ICC present and active
dwLength = 00000003h
abData = (NAD,PCB,LEN=1) = prologue field S(WTX request).

Bulk - IN

RDR_to_PC_DataBlock
bSlot = 01h, bSeq = 47h
bStatus=00h=ICC present and active
dwLength = length of abData field
abData = Information field (1 byte) and Epilogue field of SBlock

Bulk - IN

RDR_to_PC_DataBlock
bSlot = 01h, bSeq = 48h
bStatus=00h=ICC present and active
dwLength = 00000003h
abData = (NAD,PCB,LEN) = prologue field I(0,0).

Bulk - IN

RDR_to_PC_DataBlock
bSlot = 01h, bSeq = 49h
bStatus=00h=ICC present and active
dwLength = length of abData field
abData = Information field (empty) and Epilogue field.

wait up to an extra ZZ BW T times before timing out.
7. APPENDIX C “APDU Level” message exchange sequences

Note: The sequences in Appendices A-D are not exhaustive of all the possible cases. They are only meant as examples of several of the possible sequences the CCID must be able to handle.

Note: The fields shown in the messages in the sequences in Appendices A-D are only those fields applicable to that message.

Note: These sequences show exchanges with an ICC in slot bSlot=01h. This is the “second” slot of a multiple slot CCID. Exchanges with an ICC in the “first” slot of a multiple slot CCID or in the only slot of a single slot CCID would require using bSlot=00h.

Note: Buffer length = dwMaxCCIDMessageLength - 10 (header length).
Protocol T=0 sequence case 1 command

**USB HOST**

**CHIP CARD INTERFACE DEVICE**

**ICC**

---

**Bulk-OUT**

**PC_to_RDR_XfrBlock**

- bSlot = 01h
- bSeq = 0Ah
- wLevelParameter = 0000h
- bBW1 byte = 00h
- dwLength = 00000004h
- abData = (CLA, INS, P1, P2) = ICC command

**Bulk-IN**

**RDR_to_PC_DataBlock**

- bSlot = 01h
- bSeq = 0Ah
- bStatus = 00h = ICC present and active
- dwLength = 00000002h
- abData = (90h, 00h) = SW1 and SW2 status bytes.
Protocol T=0 sequence case 2 command (expect data from the ICC) with the length not accepted by the ICC.

(buffer length >= APDU length)

---

**USB HOST**

- **PC_to_RDR_XfrBlock**
  - **bSlot** = 01h, **bSeq** = 0Bh
  - **wLevelParameter** = 0000h
  - **bBWI byte** = 00h
  - **dwLength** = 00000005h
  - **abData** = (CLA, INS, P1, P2, P3) = ICC command

**CHIP CARD INTERFACE DEVICE**

- **CLA,INS,P1,P2,P3**

**ICC**

- **6C,XX**
- **INS,Data1,Data2,...,DataXX,90,00**

---

**USB HOST**

- **RDR_to_PC_DataBlock**
  - **bSlot** = 01h, **bSeq** = 0Bh
  - **bStatus** = 00h = ICC present and active
  - **dwLength** = XX+2
  - **abData** = (Data(XX), 90h, 00h) = ICC’s data and SW1 and SW2 status bytes.
Protocol T=0  sequence case 2 command (expect data from the ICC),
Procedure byte=not INS, buffer length >= APDU length

**USB HOST**

**CHIP CARD INTERFACE DEVICE**

**ICC**

---

**PC_to_RDR_XfrBlock**
- bSlot = 01h, bSeq = 0Ch
- wLevelParameter = 0000h
- bBW1 byte = 00h
- dwLength = 00000005h
- abData = (CLA, INS, P1, P2, P3) = ICC command

---

**RDR_to_PC_DataBlock**
- bSlot = 01h, bSeq = 0Ch
- bStatus = 00h = ICC present and active
- dwLength = YY+2
- abData = (Data(YY), 90h, 00h) = ICC’s data and SW1 and SW2 status bytes

---

**CLA, INS, P1, P2, P3**

---

6C, YY

Not INS, Data1

Not INS, Data2

Not INS, DataYY, 90, 00

---

7. APPENDIX C “APDU Level” message exchange sequences
Protocol T=0 sequence case 3 command (send data to the ICC),

(buffer length >= APDU length)

<table>
<thead>
<tr>
<th>USB HOST</th>
<th>CHIP CARD INTERFACE DEVICE</th>
<th>ICC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bulk-OUT</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>PC_to_RDR_XfrBlock</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bSlot = 01h, bSeq = 0Dh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wLevelParameter = 0000h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bBW1 byte=00h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dwLength = Lc+5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>abData = (CLA,INS,P1,P2,Lc,Data(Lc)) = ICC command and data</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| | USB HOST | CHIP CARD INTERFACE DEVICE | ICC |
| | | | |
| **Bulk-IN** | | | |
| **RDR_to_PC_DataBlock** | | |
| bSlot = 01h, bSeq = 0Dh | | |
| bStatus=00h=ICC present and active | | |
| dwLength = 00000002h | | |
| abData = (90h,00h) = ICC’s SW1 and SW2 status bytes. | | |

(7. APPENDIX C “APDU Level” message exchange sequences)
Protocol T=0 sequence case 3 command (send data to the ICC)

ICC request waiting time extension, buffer length >= APDU length

---

**USB HOST**

**CHIP CARD INTERFACE**

**ICC**

---

**PC_to_RDR_XfrBlock**
- bSlot = 01h, bSeq = 0Eh
- wLevelParameter = 0000h
- bBW1 byte = 00h
- dwLength = Lc + 5
- abData = (CLA, INS, P1, P2, Lc, Data(Lc)) = ICC command and data

**RDR_to_PC_DataBlock**
- bSlot = 01h, bSeq = 0Eh
- bStatus = 00h = ICC present and active
- Data(Lc)

---

**PC_to_RDR_XfrBlock**
- bSlot = 01h, bSeq = 0Eh
- wLevelParameter = 0000h
- bBW1 byte = 00h
- dwLength = Lc + 5
- abData = (CLA, INS, P1, P2, Lc, Data(Lc)) = ICC command and data

**RDR_to_PC_DataBlock**
- bSlot = 01h, bSeq = 0Eh
- bStatus = 00h = ICC present and active
- bError = 01h
- dwLength = 00000000h
- abData is absent

---

**PC_to_RDR_XfrBlock**
- bSlot = 01h, bSeq = 0Eh
- wLevelParameter = 0000h
- bBW1 byte = 00h
- dwLength = Lc + 5
- abData = (CLA, INS, P1, P2, Lc, Data(Lc)) = ICC command and data

**RDR_to_PC_DataBlock**
- bSlot = 01h, bSeq = 0Eh
- bStatus = 00h = ICC present and active
- dwLength = 00000000h
- abData = (90h, 00h) = ICC’s SW1 and SW2 status bytes.
Protocol T=0 sequence case 4 command
(send data to the ICC, receive data from the ICC),
(buffer length >= APDU length)

* The CLA (class) byte for the Get Response command can be one of several values. When an ICC is newly inserted into a slot, the CCID will use the value of the bClassGetResponse field of the Class Descriptor until a PC_to_RDR_T0APDU command message is received which changes the bClassGetResponse value for that slot.
Protocol T=0 sequence case 4 command 
(send data to the ICC, receive data from the ICC),
(buffer length < APDU length)

* The CLA (class) byte for the Envelope command can be one of several values. When an ICC is newly inserted into a slot, the CCID will use the value of the bClassEnvelope field of the Class Descriptor until a PC_to_RDR_T0APDU command message is received which changes the bClassEnvelope value for that slot.

7. APPENDIX C “APDU Level” message exchange sequences
Protocol T=0 sequence case 4 command

(send data to the ICC, receive data from the ICC),

(buffer length < APDU length)

<table>
<thead>
<tr>
<th>USB HOST</th>
<th>CHIP CARD INTERFACE DEVICE</th>
<th>ICC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Continuation</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Get response</strong></td>
<td><em><em>CLA</em>,C0,00,00,00h</em>*</td>
<td></td>
</tr>
</tbody>
</table>
| **Get response exchange repeated until buffer full** | | **C0**

**Bulk - IN**

**RDR_to_PC_DataBlock**

bSlot = 01h, bSeq = 15h
bStatus=00h=ICC present and active
bChainParameter = 01h = begin response APDU and continue
dwLength = buffer length
abData = first part of response APDU

255 bytes of response APDU, 61,xx

**Bulk - OUT**

**PC_to_RDR_XfrBlock**

bSlot = 01h, bSeq = 16h
wLevelParameter = 0010h = continuation of response APDU expected
bBW I byte=00h
dwLength = 00000000h
abData is absent

**Get response exchange repeated until buffer full**

**Bulk - IN**

**RDR_to_PC_DataBlock**

bSlot = 01h, bSeq = 16h
bStatus=00h=ICC present and active
bChainParameter = 03h = Response APDU continuation
dwLength = buffer length until the last partial buffer
abData=next part of the response APDU

**Repeat 3 to 4 until all of the response APDU is completed,** (SW1,SW2 =90,00).
The last Bulk-IN sent has
bChainParameter = 02 and data contains the last part of the response APDU and the SW1,SW2 status bytes

*The CLA (class) byte for the Get Response command can be one of several values.*

*When an ICC is newly inserted into a slot, the CCID will use the value of the bClassGetResponse field of the Class Descriptor until a PC_to_RDR_T0APDU command message is received which changes the bClassGetResponse value for that slot.*
Protocol $T=1$ sequence case 4 command

(send data to and receive data from the ICC)

ICC request waiting time extension, buffer length $\geq$ APDU length

---

### USB HOST

**PC_to_RDR_XfrBlock**
- $bSlot = 01h$, $bSeq = 0Ah$
- $wLevelParameter = 0000h$
- $bBW1\; byte = 00h$
- $dwLength = Lc + 6$
- $abData = (CLA,INS,P1,P2,Lc,Data(Lc), Le) =$ ICC command and data

### CHIP CARD INTERFACE DEVICE

**RDR_to_PC_DataBlock**
- $bSlot = 01h$, $bSeq = 0Ah$
- $bStatus = 00h =$ ICC present and active
- $bError=03h$
- $dwLength = 00000000h$
- $abData = (Data(Le),90h,00h) =$ ICC's data and SW1 and SW2 status bytes

---

### ICC

**S(WTX request,3)**

**S(WTX response,3)**

**I(0,0)=$NAD,PCB,LEN,CLA,INS,P1,\ P2,Lc,Data(Lc),Lc,LRC**

---

7. APPENDIX C “APDU Level” message exchange sequences
Protocol T=1 sequence case 4 command

(send data to and receive data from the ICC)

buffer length < APDU length

---

**USB HOST**

**CHIP CARD INTERFACE DEVICE**

**ICC**

---

**Bulk-OUT**

PC_to_RDR_XfrBlock

- bSlot = 01h, bSeq = 2Ah
- wLevelParameter = 0001h = APDU to continue
- bBWl byte = 00h
- dwLength = buffer length
- abData = (CLA,INS,P1,P2,Lc, Data(buffer length-7)) = ICC command and data (NOTE: Lc is 3 bytes long).

---

**Bulk-OUT**

PC_to_RDR_XfrBlock

- bSlot = 01h, bSeq = 2Bh
- wLevelParameter = 0003h = APDU continues and another block follows
- bBWl byte = 00h
- dwLength = buffer length until the last partial buffer
- abData = next part of the APDU

---

**1**

I-block exchanges repeated until buffer empty

**R(1)**

---

**Bulk-OUT**

RDR_to_PC_DataBlock

- bSlot = 01h, bSeq = 2Ah
- bStatus = 00h = ICC present and active
- bChainParameter = 10h = Continue
- dwLength = 00000000h
- abData = absent

---

**2**

Repeat 1 to 2 until wLevelParameter = 0002h, APDU ends and data field contains last part of APDU.

I-block exchanges repeated until last part of APDU command is sent.

---

Continued on next page

---

7. APPENDIX C “APDU Level” message exchange sequences
Protocol T=1 sequence case 4 command

(send data to and receive data from the ICC)

buffer length < APDU length

Continuation

<table>
<thead>
<tr>
<th>USB HOST</th>
<th>CHIP CARD INTERFACE DEVICE</th>
<th>ICC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Continued from previous page

I(0,1)=NAD,PCB,FEh,first 254 APDU response bytes,LRC

R(1)

I-block exchanges from ICC repeated until buffer full

Bulk-IN

RDR_to_PC_DataBlock

bSlot = 01h, bSeq = 31h
bStatus=00h=ICC present and active
bChainParameter = 01h = begin response APDU and continue
dwLength = buffer length
abData = first part of response APDU

Bulk-OUT

PC_to_RDR_XfrBlock

bSlot = 01h, bSeq = 32h
wLevelParameter = 0010h = continuation of response APDU is expected
bBW1 byte=00h
dwLength = 00000000h
abData is absent

I-block exchanges from ICC repeated until buffer full

Bulk-IN

RDR_to_PC_DataBlock

bSlot = 01h, bSeq = 32h
bStatus=00h=ICC present and active
bChainParameter = 03h = response APDU continuation
dwLength = buffer length until the last partial buffer
abData = next part of the response APDU

Repeat 3 to 4 until all of the response APDU is completed, R(x,0), the last Bulk-IN sent has the
bChainParameter = 02h response
APDU end, and the data contains the last part of the response APDU and the SW1,SW2 status bytes

7. APPENDIX C “APDU Level” message exchange sequences
8. APPENDIX D “TPDU Level” message exchange sequences

Note: The sequences in Appendices A-D are not exhaustive of all the possible cases. They are only meant as examples of several of the possible sequences the CCID must be able to handle.

Note: The fields shown in the messages in the sequences in Appendices A-D are only those fields applicable to that message.

Note: These sequences show exchanges with an ICC in slot bSlot=01h. This is the "second" slot of a multiple slot CCID. Exchanges with an ICC in the “first” slot of a multiple slot CCID or in the only slot of a single slot CCID would require using bSlot=00h.
Protocol $T=0$ sequence ‘pseudo TPDU’ PPS Exchange

**USB HOST**

**CHIP CARD INTERFACE DEVICE**

**ICC**

---

**PC_to_RDR_XfrBlock**
- $bSlot = 01h$, $bSeq = 33h$
- $wLevelParameter = 0000h$
- $b BWI byte = 00h$
- $dwLength = length of abData field$
- $abData = (FFh,PPS0,...,PCK) = PPS$ request (Protocol and Parameter Selection)

**RDR_to_PC_DataBlock**
- $bSlot = 01h$, $bSeq = 33h$
- $bStatus=00h$ = ICC present and active
- $dwLength = length of abData field$
- $abData = (FFh,PPS0_R,...,PCK_R) = PPS$ response

---

**Bulk-OUT**

**PPS Request:**
- FFh, PPS0,..., PCK

**PPS Response:**
- FFh, PPS0_R,..., PCK_R

---

8. APPENDIX D “TPDU Level” message exchange sequences
Protocol T=0 sequence case 4 command

(send data to and receive data from the ICC)

short APDU transportation

APDU=CLA, INS, P1, P2, Lc, Data(Lc), Le

8. APPENDIX D “TPDU Level” message exchange sequences
Protocol T=0  sequence case 1 command

No data send, no data expected

USB HOST

<table>
<thead>
<tr>
<th>Protocol T=0 sequence case 1 command</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>USB HOST</strong></td>
</tr>
<tr>
<td><strong>CHIP CARD INTERFACE DEVICE</strong></td>
</tr>
<tr>
<td><strong>ICC</strong></td>
</tr>
<tr>
<td><strong>Bulk-OUT</strong></td>
</tr>
<tr>
<td><strong>PC_to_RDR_XfrBlock</strong></td>
</tr>
<tr>
<td>bSlot = 01h, bSeq = 36h</td>
</tr>
<tr>
<td>wLevelParameter = 0000h</td>
</tr>
<tr>
<td>bBW1byte = 00h</td>
</tr>
<tr>
<td>dwLength = 4</td>
</tr>
<tr>
<td>abData = (CLA,INS,P1,P2) = ICC command</td>
</tr>
<tr>
<td><strong>Bulk-IN</strong></td>
</tr>
<tr>
<td><strong>RDR_to_PC_DataBlock</strong></td>
</tr>
<tr>
<td>bSlot = 01h, bSeq = 36h</td>
</tr>
<tr>
<td>bStatus = 00h=ICC present and active</td>
</tr>
<tr>
<td>bError = 00h</td>
</tr>
<tr>
<td>dwLength = 00000002h</td>
</tr>
<tr>
<td>abData = (SW1,SW2) = ICC’s status bytes</td>
</tr>
</tbody>
</table>

8. APPENDIX D “TPDU Level” message exchange sequences
Protocol T=0 sequence case 2 command

(expect data from the ICC)

ICC request waiting time extension

USB HOST

<table>
<thead>
<tr>
<th>Protocol T=0 sequence case 2 command</th>
<th>USB HOST</th>
</tr>
</thead>
<tbody>
<tr>
<td>(expect data from the ICC)</td>
<td></td>
</tr>
<tr>
<td>ICC request waiting time extension</td>
<td></td>
</tr>
</tbody>
</table>

CHIP CARD INTERFACE

<table>
<thead>
<tr>
<th>Protocol T=0 sequence case 2 command</th>
<th>CHIP CARD INTERFACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>(expect data from the ICC)</td>
<td></td>
</tr>
<tr>
<td>ICC request waiting time extension</td>
<td></td>
</tr>
</tbody>
</table>

ICC

<table>
<thead>
<tr>
<th>Protocol T=0 sequence case 2 command</th>
<th>ICC</th>
</tr>
</thead>
<tbody>
<tr>
<td>(expect data from the ICC)</td>
<td></td>
</tr>
<tr>
<td>ICC request waiting time extension</td>
<td></td>
</tr>
</tbody>
</table>

8. APPENDIX D “TPDU Level” message exchange sequences
Protocol T=1 sequence without chaining

APDU=CLA, INS, P1, P2, Lc, Data(Lc), Le

8. APPENDIX D “TPDU Level” message exchange sequences
Protocol T=1 sequence with chaining

**USB HOST**

- **PC_to_RDR_XfrBlock**
  - bSlot = 01h, bSeq = 39h
  - wLevelParameter = 0000h
  - bBW I byte=00h
  - dwLength = length of abData field
  - abData = I-block I(0,0)

- **PC_to_RDR_XfrBlock**
  - bSlot = 01h, bSeq = 3Ah
  - wLevelParameter = 0000h
  - bBW I byte=00h
  - dwLength = length of abData field
  - abData = I-block I(1,0)

- **PC_to_RDR_XfrBlock**
  - bSlot = 01h, bSeq = 3Bh
  - wLevelParameter = 0000h
  - bBW I byte=00h
  - dwLength = length of abData field
  - abData = R-block R(1)

**CHIP CARD INTERFACE DEVICE**

- **RDR_to_PC_DataBlock**
  - bSlot= 01h, bSeq = 39h
  - bStatus=00h=ICC present and active
  - dwLength = length of abData field
  - abData = R-block R(1)

- **RDR_to_PC_DataBlock**
  - bSlot= 01h, bSeq = 3Ah
  - bStatus=00h=ICC present and active
  - dwLength = length of abData field
  - abData = I-block response I(0,1)

- **RDR_to_PC_DataBlock**
  - bSlot= 01h, bSeq = 3Bh
  - bStatus=00h=ICC present and active
  - dwLength = length of abData field
  - abData = I-block response I(1,0)

**ICC**

- **Bulk-IN**
  - R(1)

- **Bulk-IN**
  - I(0,0)

- **Bulk-IN**
  - I(1,0)

- **Bulk-IN**
  - I(0,1)

8. APPENDIX D “TPDU Level” message exchange sequences
Protocol T=1 sequence with waiting time extension request

<table>
<thead>
<tr>
<th>USB HOST</th>
<th>CHIP CARD INTERFACE DEVICE</th>
<th>ICC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulk-OUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>PC_to_RDR_XferBlock</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bSlot = 01h, bSeq = 3Dh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wLevelParameter = 0000h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bBW I byte = 00h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dwLength = length of abData field</td>
<td></td>
<td></td>
</tr>
<tr>
<td>abData = I-block I(0,0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bulk-OUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>PC_to_RDR_XferBlock</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bSlot = 01h, bSeq = 3Eh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wLevelParameter = 0000h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bBW I byte = 02h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dwLength = length of abData field</td>
<td></td>
<td></td>
</tr>
<tr>
<td>abData = S-block response S(WTX response,2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bulk-OUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>PC_to_RDR_XferBlock</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bSlot = 01h, bSeq = 3Eh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wLevelParameter = 0000h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bBW I byte = 02h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dwLength = length of abData field</td>
<td></td>
<td></td>
</tr>
<tr>
<td>abData = I-block I(0,0)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8. APPENDIX D “TPDU Level” message exchange sequences
9. APPENDIX E  PIN Management Examples

9.1 PIN Verification

Examples of format coding for PIN verification:

<table>
<thead>
<tr>
<th>Example</th>
<th>Format</th>
<th>Type</th>
<th>Pad</th>
<th>PIN</th>
<th>Bit</th>
<th>Justification</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>BCD</td>
<td>ASCII</td>
<td>Binary</td>
<td>Length</td>
<td>Shift</td>
<td>Left</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>*</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td>*</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>*</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>*</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>*</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

9.1.1 Example 1: PIN uses a binary format conversion
- PIN entered is 12345678 (min and max sizes are 8 digits)
- Left justification
- No messages
- The CCID sends to the ICC the command

```
CLA INS P1 P2 Lc 01 02 03 04 05 06 07 08
0000

bTimeOut 00
bmFormatString 0000 0000 b
bmPINBlockString 0000 1000 b
bmPINLengthFormat 00h
wPINMaxExtraDigit 0808h
bEntryValidationCondition 01h
bNumberMessage 00h
wLangId 0409h
abData CLA INS P1 P2 08
0000000000000000
```

Preliminary data: 00h 00h 00h 00h 00h 00h 00h 00h is part of abData

9.1.1.1 Initial data mapping by the device

Data 1 = 00h
```
0 0 0 0 0 0 0
```

Data 2 = 00h
```
0 0 0 0 0 0 0
```

Data 3 = 00h
```
0 0 0 0 0 0 0
```
9. APPENDIX E  PIN Management Examples

9.1.1.2 After key entry + Binary conversion + left justification

<table>
<thead>
<tr>
<th>Data 1 = 01h</th>
<th>Data 2 = 02h</th>
<th>Data 3 = 03h</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 0 1</td>
<td>0 0 0 0 0 0 1 0</td>
<td>0 0 0 0 0 0 1 1</td>
</tr>
</tbody>
</table>

9.1.2 Example 2: **PIN uses a shift rotation format conversion.**
- PIN entered is 4330 (min and max sizes are 4 digits)
- Left justification
- 1 message “ENTER PIN”
- The CCID sends to the ICC the command

```
CLA INS P1 P2 Lc 10 CC 3F FF
```

- **bTimeOut**: 00h
- **bmFormatString**: 0 0010 0 01 b
- **bmPINBlockString**: 0000 0100 b
- **bmPINLengthFormat**: 00h
- **wPINMaxExtraDigit**: 0404h
- **bEntryValidationCondition**: 01h
- **bNumberMessage**: 01h
- **wLangId**: 040Ch
- **bMsgIndex**: 0h
- **abData**: Cla INS P1 P2 04 00003FFF

Preliminary data: 00h 00h 3Fh FFh is part of abData
9.1.2.1 Initial data mapping

Data 1 = 00h  Data 2 = 00h  Data 3 = 3Fh

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

Sys.  Secret code byte 1 (2digits).  Secret code byte 2 (2 digits)  automatic

Data 4 = FFh

1 1 1 1 1 1 1

Padding operation

9.1.2.2 After key entry + left justification

Data 1 = 10h  Data 2 = CCh  Data 3 = 3Fh

| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

PIN: 4 and 3  PIN: 3 and 0

Data 4 = FFh

1 1 1 1 1 1 1 1
9.1.3 Example 3: **PIN uses a BCD format conversion with PIN length insertion**
- PIN entered is 1234 (min size = 4 and max size = 12 digits)
- Left justification
- The first data contains a control field (0010) and the PIN length.
- No messages
- The CCID sends to the ICC the command

```
CLA  INS  P1  P2  Lc  24  12  34  FF  FF  FF  FF
bTimeOut 00
bmFormatString 1 0001 0 01 b
b System unit = byte
PIN position in the frame = 1 byte
PIN justification left
BCD format
bmPINBlockString 0100 0111 b
PIN length size: 4 bits
Length PIN = 7 bytes
bmPINLengthFormat 000 0 0100 b
System bit units is bit
PIN length is at the 4th position bit
wPINMaxExtraDigit 040Ch
Min=4  Max ~12 digits
bEntryValidationCondition 03h
Max size reach or Validation key pressed
bNumberMessage 00h
No message
wLangId 0C0Ah
Language is Spanish
abData CLA INS P1 P2 08
20FFFFFFFFFFFFFFF
```

Preliminary data: 20h FFh FFh FFh FFh FFh FFh FFh is part of abData

9.1.3.1 Initial data mapping by the device

<table>
<thead>
<tr>
<th>Data 1 = 20h</th>
<th>Data 2 = FFh</th>
<th>Data 3 = FFh</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 0 0 0 0 0</td>
<td>1 1 1 1 1 1 1 1</td>
<td>1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>Control field</td>
<td>PIN length</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data 4 = FFh</th>
<th>Data 5 = FFh</th>
<th>Data 6 = FFh</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 1 1 1 1 1</td>
<td>1 1 1 1 1 1 1 1</td>
<td>1 1 1 1 1 1 1 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data 7 = FFh</th>
<th>Data 8 = FFh</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 1 1 1 1 1</td>
<td>1 1 1 1 1 1 1 1</td>
</tr>
</tbody>
</table>

9.1.3.2 After key entry + left justification

<table>
<thead>
<tr>
<th>Data 1 = 24h</th>
<th>Data 2 = 12h</th>
<th>Data 3 = 34h</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 0 0 0 0 0</td>
<td>1 1 1 1 1 1 1 1</td>
<td>1 1 1 1 1 1 1 1</td>
</tr>
</tbody>
</table>

9. APPENDIX E PIN Management Examples
<table>
<thead>
<tr>
<th>PIN length</th>
<th>Data 4 = FFh</th>
<th>Data 5 = FFh</th>
<th>Data 6 = FFh</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 0 0 1 0 0</td>
<td>1 1 1 1 1 1 1 1</td>
<td>1 1 1 1 1 1 1 1</td>
<td>1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>Data 7 = FFh</td>
<td>Data 8 = FFh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 1 1 1 1 1 1</td>
<td>1 1 1 1 1 1 1 1</td>
<td>1 1 1 1 1 1 1 1</td>
<td></td>
</tr>
</tbody>
</table>
9.1.4 Example 4: PIN uses BCD format conversion with right justification and a control field.

- PIN entered is 13579 (min size = 4 and max size = 8 digits)
- Right justification. The personal code contains less than 8 digits; therefore, the most significant digits of the eight-digit code must be filled with zeroes.
- The frame integrates a specific control field “01” before the PIN conversion.
- No messages
- The CCID sends to the ICC the command

```
CLA  INS P1 P2 Lc  01  00  01  35  79
```

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bTimeOut</td>
<td>00h</td>
</tr>
<tr>
<td>bmFormatString</td>
<td>0001 01b</td>
</tr>
<tr>
<td>bmPINBlockString</td>
<td>0000 0100 b</td>
</tr>
<tr>
<td>bmPINLengthFormat</td>
<td>00h</td>
</tr>
<tr>
<td>wPINMaxExtraDigit</td>
<td>0408h</td>
</tr>
<tr>
<td>bEntryValidationCondition</td>
<td>03h</td>
</tr>
<tr>
<td>bNumberMessage</td>
<td>00h</td>
</tr>
<tr>
<td>wLangId</td>
<td>0410h</td>
</tr>
<tr>
<td>abData</td>
<td>CLA INS P1 P2 05 01000000</td>
</tr>
</tbody>
</table>

Default time out

System unit = byte

PIN position in the frame = 1 byte
Justification right

BCD format

No PIN length management
Length PIN = 4 bytes

Max size reach or Validation key pressed

Language is Italian

APDU Command

Preliminary data: 01h 00h 00h 00h 00h is part of abData

9.1.4.1 Initial data mapping by the device

Data 1 = 01h

```
0 0 0 0 0 0 0 1
```

Data 2 = 00h

```
0 0 0 0 0 0 0 0
```

Data 3 = 00h

```
0 0 0 0 0 0 0 0
```

Data 4 = 00h

```
0 0 0 0 0 0 0 0
```

Data 5 = 00h

```
0 0 0 0 0 0 0 0
```

9.1.4.2 After key entry + right justification

Data 1 = 01h

```
0 0 0 0 0 0 0 1
```

Data 2 = 00h

```
0 0 0 0 0 0 0 0
```

Data 3 = 01h

```
0 0 0 0 0 0 0 1
```

Data 4 = 35h

```
0 0 0 0 0 0 0 0
```

Data 5 = 79h

```
0 0 0 0 0 0 0 0
```
9.1.5 Example 5: **PIN uses an ASCII format conversion with padding.**

- PIN entered is 1357 (min size = 4 and max size = 8 digits)
- Left justification
- Default display behavior for the CCID
- The CCID sends to the ICC the command

<table>
<thead>
<tr>
<th>CLA</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>Lc</th>
<th>31</th>
<th>33</th>
<th>35</th>
<th>37</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
</tbody>
</table>

- bTimeOut: 00h
- bmFormatString: 0000 010b
  - System unit = bit
  - PIN position in the frame = 0
  - Justification left
  - ASCII format
- bmPINBlockString: 0000 1000b
  - No PIN length management
  - Length PIN = 8 bytes
- bmPINLengthFormat: 0
  - No PIN length management
- wPINMaxExtraDigit: 0408h
  - Min = 4 max = 8 digits
- bEntryValidationCondition: 03h
  - Max size reached or Validation key pressed
- bNumberMessage: FFh
  - CCID default messages
- wlangId: 041Dh
  - Language is Swedish
- abData: CLA INS P1 P2 08 FF FFFF FFFF FFFF FFFF

Preliminary data: FFh FFh FFh FFh FFh FFh FFh FFh is part of abData

### 9.1.5.1 Initial data mapping

<table>
<thead>
<tr>
<th>Data 1 = FFh</th>
<th>Data 2 = FFh</th>
<th>Data 3 = FFh</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 1 1 1 1</td>
<td>1 1 1 1 1 1 1</td>
<td>1 1 1 1 1 1 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data 4 = FFh</th>
<th>Data 5 = FFh</th>
<th>Data 6 = FFh</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 1 1 1 1</td>
<td>1 1 1 1 1 1 1</td>
<td>1 1 1 1 1 1 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data 7 = FFh</th>
<th>Data 8 = FFh</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 1 1 1 1</td>
<td>1 1 1 1 1 1 1</td>
</tr>
</tbody>
</table>

### 9.1.5.2 PIN + left justification

<table>
<thead>
<tr>
<th>Data 1 = 31h</th>
<th>Data 2 = 33h</th>
<th>Data 3 = 35h</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 1 0 0 1</td>
<td>0 0 1 1 0 0 1</td>
<td>0 0 1 1 0 1 0 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data 4 = 37h</th>
<th>Data 5 = FFh</th>
<th>Data 6 = FFh</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 1 0 0 1</td>
<td>0 0 1 1 0 1 0 1</td>
<td></td>
</tr>
</tbody>
</table>
Padding with F character

Data 7 = FFh

Data 8 = FFh
9.2 PIN Modification

<table>
<thead>
<tr>
<th>Example</th>
<th>Format</th>
<th>Type</th>
<th>Pad</th>
<th>PIN Bit</th>
<th>Justification</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>BCD</td>
<td>ASCII</td>
<td>Binary</td>
<td>Length</td>
<td>Shift</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

9.2.1 Example 1: Change PIN ASCII format (8-byte long).
- Confirm PIN entry option is off.
- Two string messages
  - Message 1 for the old PIN presentation
  - Message 2 for the new PIN presentation
- New PIN value is “9876”
- Old PIN value is “1357”
- The CCID sends to the ICC two commands
  - CLA INS P1 P2 Length 31h 33h 35h 37h FFh FFh FFh FFh FFh for the old PIN presentation
  - CLA INS P1 P2 Length 39h 38h 37h 36h FFh FFh FFh FFh for the new PIN presentation

As a result, the HOST will have to send first a PIN verification command in order to request the user to enter his old PIN and to present it to the ICC. Then the HOST will send a Modify PIN command in order to request the user to enter his new PIN and to present it to the ICC.

PIN verification command:

<table>
<thead>
<tr>
<th>bTimeout</th>
<th>00h</th>
<th>Default time out</th>
</tr>
</thead>
<tbody>
<tr>
<td>bmFormatString</td>
<td>00000 0 10 b</td>
<td>System unit = bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No number of system unit PIN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Justification left</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASCII format</td>
</tr>
<tr>
<td>bmPINBlockString</td>
<td>0000 1000 b</td>
<td>No PIN length management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Length PIN = 8 bytes</td>
</tr>
<tr>
<td>bmPINLengthFormat</td>
<td>00h</td>
<td>No PIN length management</td>
</tr>
<tr>
<td>wPINMaxExtraDigit</td>
<td>0408h</td>
<td>Min=4 max = 8 digits</td>
</tr>
<tr>
<td>bEntryValidationCondition</td>
<td>03h</td>
<td>Max size reach or Validation key pressed</td>
</tr>
<tr>
<td>wLangId</td>
<td>0413h</td>
<td>Language is Dutch</td>
</tr>
<tr>
<td>abData</td>
<td>CLA INS1 P1 P2 08 FFFFFFFFFFFFFFFF</td>
<td>APDU Command for old PIN presentation</td>
</tr>
</tbody>
</table>

Modify PIN Command:

<table>
<thead>
<tr>
<th>bTimeout</th>
<th>00h</th>
<th>Default time out</th>
</tr>
</thead>
</table>

9. APPENDIX E  PIN Management Examples
| bmFormatString       | 0000 010 b | System unit = bit  
|                      |           | No number of system unit PIN  
|                      |           | Justification left  
|                      |           | ASCII format  
| bmPINBlockString     | 0000 1000 b | No PIN length management  
|                      |           | Length PIN = 8 bytes  
| bmPINLengthFormat    | 00h       | No PIN length management  
| bInsertionOffsetOld  | 00h       | Position of the old PIN  
| bInsertionOffsetNew  | 08h       | Position of the new PIN  
| wPINMaxExtraDigit    | 0408h     | Min=4 max = 8 digits  
| bConfirmPIN          | 00h       | The device does not manage the PIN confirmation.  
| bEntryValidationCondition | 03h | Max size reach or Validation key pressed  
| bNumberMessage       | 01h       | Two messages  
| wLangId              | 0413h     | Language is Dutch  
| bMsgIndex1           | 01h       | Message Index for ENTER NEW PIN  
| abPINApdu            | CLA INS2 P1 P2 08 FFFFFFFFFFFF | APDU command for new PIN presentation  

9. APPENDIX E PIN Management Examples
### 9.2.2 Example 2: PIN uses an ASCII format conversion with PIN length management.

- Old PIN entered is 1234
- New PIN entered is 56789
- Left justification
- Confirm PIN entry option On
- Three internal messages
  - Message 1 for the old PIN presentation: “Enter OLD PIN”
  - Message 2 for the new PIN presentation: “Enter NEW PIN”
  - Message 3 for the new PIN confirmation: “CONFIRM PIN”
- The CCID sends to the ICC the command

```
CLA INS P1 P2 Lc 24 31 32 33 34 FF FF FF 25 35 36 37 38 39 FF FF  
```

<table>
<thead>
<tr>
<th>bTimeOut</th>
<th>00h</th>
<th>Default time out</th>
</tr>
</thead>
</table>
| bmFormatString    | 1 0001 0 10 b | System unit = byte  
|                   |      | PIN position in the frame =1  
|                   |      | Justification left  
|                   |      | ASCII format                      |
| bmPINBlockString  | 0100 0111 b | Pin length stored within 4 bits  
|                   |      | Length PIN = 7 bytes               |
| bmPINLengthFormat | 000 0 0100 b | PIN length management used  
|                   |      | System bit units is bit  
|                   |      | PIN length is at the 4th position bit |
| bInsertionOffsetOld | 00h | Position of the old PIN          |
| bInsertionOffsetNew | 08h | Position of the new PIN          |
| wPINMaxExtraDigit | 0407h | Min=4 max = 7 digits         |
| bConfirmPIN       | 01h  | The device must manage the PIN confirmation |
| bEntryValidationCondition | 03h | Max size reach or Validation key pressed |
| bNumberMessage    | 03h  | Three messages                      |
| wLangId           | 0411h | Language is Japanese              |
| bMsgIndex 1       | 00h  | Message Index for ENTER PIN        |
| bMsgIndex 2       | 01h  | Message Index for ENTER NEW PIN    |
| bMsgIndex 3       | 02h  | Message Index for CONFIRM PIN      |
| abPINApdu         | CLA INS P1 P2 10  
|                   |  20FFFFFFF  
|                   |  20FFFFFFF  |

Preliminary data: 20h FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh FFh

### 9.2.2.1 Initial data mapping OldPIN = NewPIN

| Data 1 = 20h | Data 2 = FFh | Data 3 = FFh |

9. APPENDIX E PIN Management Examples
9.2.2.2 PIN + left justification

OldPIN Data 1 = 24h

0 0 1 0 0 1 0 0

OldPIN Data 2 = 31h

0 0 1 1 0 0 0 1

OldPIN Data 3 = 32h

0 0 1 1 0 0 1 0

OldPIN Data 4 = 33h

0 0 1 1 0 0 1 1

OldPIN Data 5 = 34h

0 0 1 1 0 1 0 0

OldPIN Data 6 = FFh

1 1 1 1 1 1 1 1

OldPIN Data 7 = FFh

1 1 1 1 1 1 1 1

OldPIN Data 8 = FFh

1 1 1 1 1 1 1 1

NewPIN Data 1 = 25h

0 0 1 0 0 1 0 1

NewPIN Data 2 = 35h

0 0 1 1 0 1 0 1

NewPIN Data 3 = 36h

0 0 1 1 0 1 1 0

NewPIN Data 4 = 37h

0 0 1 1 0 1 1 1

NewPIN Data 5 = 38h

0 0 1 1 0 0 0 0

NewPIN Data 6 = 39h

0 0 1 1 0 0 1 0

NewPIN Data 7 = FFh

1 1 1 1 1 1 1 1

NewPIN Data 8 = FFh

1 1 1 1 1 1 1 1

The parameters bInsertionOffsetOld, bInsertionOffsetNew, bmPINLengthFormat and bmFormatString provide the right position of both PIN (old and new) in the APDU command. Indeed:
9.2.2.3 First operation: PIN conversion

\[
\begin{array}{cccccccccccc}
0 & 20 & 31 & 32 & 33 & 34 & FF & FF & FF & 20 & 35 & 36 & 37 & 38 & 39 & FF & FF \\
\end{array}
\]

\[b_{\text{InsertionOffsetOld}}(0h) + \text{bmFormatString}(b_6 - b_3 = 01h) = \text{position } 1\]
\[b_{\text{InsertionOffsetNew}}(08h) + \text{bmFormatString}(b_6 - b_3 = 01h) = \text{position } 9\]

9.2.2.4 Second operation: APDU command format + PIN length insertion

The position of the old PIN’s length is:
\[b_{\text{InsertionOffsetOld}}(00h) + \text{bmPINLengthFormat}(4 \text{ bits}) = 0.5 \text{ bytes}\]

The position of the new PIN’s length is:
\[b_{\text{InsertionOffsetNew}}(08h) + \text{bmPINLengthFormat}(4 \text{ bits}) = 8.5 \text{ bytes}\]
9.2.3 Character Level, Protocol T=0, sequence for PIN verification

<table>
<thead>
<tr>
<th>USB HOST</th>
<th>CHIP CARD INTERFACE DEVICE</th>
<th>ICC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PC_to_RDR_Secure</strong></td>
<td><strong>PC_to_RDR_Secure</strong></td>
<td><strong>RDR_to_PC_DataBlock</strong></td>
</tr>
<tr>
<td>bSlot = 01h, bSeq = 0Ch</td>
<td>bSlot = 01h, bSeq = 0Ch</td>
<td>bSlot = 01h, bSeq = 0Ch</td>
</tr>
<tr>
<td>wLevelParameter = 001h</td>
<td>wLevelParameter = 0001h</td>
<td>wLevelParameter = 0001h</td>
</tr>
<tr>
<td>bBW I byte = 00h</td>
<td>bBW I byte = 00h</td>
<td>bStatus = 00h = ICC present and active</td>
</tr>
<tr>
<td>dwLength = length of data structure</td>
<td>dwLength = 00000000h</td>
<td>dwLength = 00000000h</td>
</tr>
<tr>
<td>abData = (CLA,INS,P1,P2,P3) = ICC command</td>
<td>abData is absent</td>
<td>abData = (CLA,INS,P1,P2,P3) = ICC command</td>
</tr>
<tr>
<td>bPINOperation = 00h (verify PIN)</td>
<td>bPINOperation = 03h (wait ICC response)</td>
<td>bPINOperation = 02h (transfer PIN)</td>
</tr>
<tr>
<td>Other fields: PIN description</td>
<td>Other fields: nothing</td>
<td>Other fields: nothing</td>
</tr>
</tbody>
</table>

Note:
In case of procedure byte equal to 60h, the host asks for 1 byte which means, for the CCID, that another procedure byte is waiting (thus the code is not sent to the ICC).

In case of procedure byte equal to INS, the host asks for 2 bytes which means, for the CCID, that it can send the code to the ICC and wait for 2 bytes from the ICC (SW1, SW2).

9. APPENDIX E PIN Management Examples
10. APPENDIX G  Application Notes

10.1 Suspend Behavior

When resuming from a USB suspend, the host/driver will assume that all ICCs have been deactivated (powered down). When the USB bus suspends, CCIDs are not required to deactivate inserted ICCs, but may do so; however, after the USB bus resumes, CCIDs must respond to the host as if all of the inserted ICCs had been deactivated and newly inserted.

After resuming, the CCID will do two things in no particular order.
1. Send the RDR_to_PC_NotifySlotChange message to inform the driver which slots have "newly inserted" cards.
2. The CCID will reactivate the ICCs only from a PC_to_RDR_IccPowerOn message from the driver or automatically if the CCID has the "automatic activation on insertion" feature. Note: When reactivating, all slot parameters initially revert back to the defaults.
11. APPENDIX H  Samples of CCID diagrams based on dwFeatures field

Note: The sequences in Appendix H are not exhaustive of all the possible cases. They are only meant as examples of several of the possible sequences the CCID must be able to handle.

11.1 dwFeatures fields used in these diagrams

FEATURE1 (0002067Eh) :
- Short APDU exchange level.
- Automatic IFSD exchange.
- NAD value other than 00h accepted.
- Automatic PPS made by the CCID (proprietary algorithm).
- Automatic baud rate change according to parameters.
- Automatic ICC clock frequency change according to parameters.
- Automatic voltage selection.
- Automatic ICC activation on insertion.
- Automatic parameters configuration based on ATR data.

FEATURE2 (000206B2h) :
- Short APDU exchange level.
- Automatic IFSD exchange.
- NAD value other than 00h accepted.
- Automatic PPS according to current parameters.
- Automatic baud rate change according to parameters.
- Automatic ICC clock frequency change according to parameters.
- Automatic parameters configuration based on ATR data.

FEATURE3 (000104B2h) :
- TPDU level exchange.
- Automatic IFSD exchange.
- Automatic PPS according to current parameters.
- Automatic baud rate change according to parameters.
- Automatic ICC clock frequency change according to parameters.
- Automatic parameters configuration based on ATR data.

FEATURE4 (00010230h) :
- TPDU level exchange
- NAD value other than 00h accepted.
- Automatic baud rate change according to parameters
- Automatic ICC clock frequency change according to parameters

FEATURE5 (00000030h) :
- Character level
- Automatic baud rate change according to parameters
- Automatic ICC clock frequency change according to parameters
FEATURE6 (00000000h) :
- No special characteristics (character level exchange, no automatism)

11.2 ICC ATRs used in these diagrams

ATR1 : - high speed offered (negotiable mode),
- only protocol T=0,
- new values for N (extra guardtime) and WI (work waiting integer),
- no voltage class announced (then class A).

ATR2 : - high speed offered (negotiable mode),
- only protocol T=0,
- new values for N (extra guardtime) and WI (work waiting integer),
- voltage class AB announced.

ATR3 : - high speed offered (negotiable mode),
- protocols T=0 and T=1,
- new values for N (extra guardtime) and WI (work waiting integer),
- new values for IFSC and CWI/BWI
- voltage class AB announced

ATR4 : - high speed (specific mode)
- only protocol T=1,
- new values for N (extra guardtime),
- new values for IFSC and CWI/BWI
- voltage class AB announced

<table>
<thead>
<tr>
<th>ATR#</th>
<th>TS</th>
<th>T0</th>
<th>TA1</th>
<th>TB1</th>
<th>TC1</th>
<th>TD1</th>
<th>TA2</th>
<th>TB2</th>
<th>TC2</th>
<th>TD2</th>
<th>TA3</th>
<th>TB3</th>
<th>TC3</th>
<th>TD3</th>
<th>TA4</th>
<th>TCK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3B</td>
<td>F0</td>
<td>18</td>
<td>00</td>
<td>02</td>
<td>40</td>
<td>05</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3B</td>
<td>F0</td>
<td>18</td>
<td>00</td>
<td>02</td>
<td>C0</td>
<td>05</td>
<td>1F</td>
<td>03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>33</td>
</tr>
<tr>
<td>3</td>
<td>3B</td>
<td>F0</td>
<td>18</td>
<td>00</td>
<td>02</td>
<td>C1</td>
<td>05</td>
<td>B1</td>
<td>40</td>
<td>38</td>
<td></td>
<td>1F</td>
<td>03</td>
<td>FB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3B</td>
<td>B0</td>
<td>18</td>
<td>00</td>
<td>D1</td>
<td>81</td>
<td>05</td>
<td>B1</td>
<td>40</td>
<td>38</td>
<td></td>
<td>1F</td>
<td>03</td>
<td>28</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
11.3 Sample diagrams of voltage management

11.3.1 CCID class AB, ATR1, Feature1

ICC class A (5v).
- Automatic ICC activation on insertion.
- Automatic voltage selection
- class AB (bVoltageSupport=03h, 3v/5v)
11.3.2 CCID class B, ATR1, Feature1

ICC class A (5v).
CCID:
- Automatic ICC activation on insertion.
- Automatic voltage selection
- class B (bVoltageSupport=02h, 3v)

---

**USB HOST**  
**CHIP CARD INTERFACE DEVICE**  
**ICC**

- **Cold Reset ICC 3v**
- **Power Down ICC**

**Interrupt - IN**
**RDR_to_PC_NotifySlotChange**
**bmSlotICCState = xxxx11xxb**

**Bulk-OUT**
**PC_to_RDR_IccPowerOn**
**bSlot = 01h, bSeq = 81h**
**dwLength = 00000000h**
**bPowerSelect = 00h**

**Bulk - IN**
**RDR_to_PC_DataBlock**
**bSlot = 01h, bSeq = 81h**
**bStatus=41h with bError =F5h and abData = ATR1**
or
**bStatus=41h with bError =FEh and abData is absent**
11.3.3 CCID class AB, ATR1, Feature2 (also Feature3, Feature4, Feature5, Feature6)

ICC class A (5v).

- no automatic feature on insertion or for voltage selection.
- class AB (bVoltageSupport=03h, 3v/5v)

### USB HOST

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC_to_RDR_ICcPowerOn</td>
<td>Bulk-OUT</td>
</tr>
<tr>
<td>bSlot = 01h, bSeq = 82h</td>
<td>RDR_to_PC_NotifySlotChange</td>
</tr>
<tr>
<td>dwLength = 00000000h</td>
<td>bmSlotICCSate = xxx11xxb</td>
</tr>
<tr>
<td>bPowerSelect = 02h</td>
<td>Cold Reset ICC 3v</td>
</tr>
<tr>
<td></td>
<td>ATR1 or ICC mute</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC_to_RDR_CardPowerOff</td>
<td>Bulk-OUT</td>
</tr>
<tr>
<td>bSlot = 01h, bSeq = 83h</td>
<td>Power Down ICC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC_to_RDR_CardPowerOn</td>
<td>Bulk-OUT</td>
</tr>
<tr>
<td>bSlot = 01h, bSeq = 84h</td>
<td>RDR_to_PC_PCSlotStatus</td>
</tr>
<tr>
<td>dwLength = 00000000h</td>
<td>bStatus=00h with bError =00h</td>
</tr>
<tr>
<td>bPowerSelect = 01h</td>
<td>Cold Reset ICC 5v</td>
</tr>
<tr>
<td></td>
<td>ATR1</td>
</tr>
</tbody>
</table>

### Icc Inserted

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDR_to_PC_DataBlock</td>
<td>Bulk - IN</td>
</tr>
<tr>
<td>bSlot = 01h, bSeq = 84h</td>
<td>bStatus=00h with bError =00h</td>
</tr>
<tr>
<td>dwLength = 00000000h</td>
<td>abData = ATR1</td>
</tr>
</tbody>
</table>

11. APPENDIX H  Samples of CCID diagrams based on dwFeatures field
11.3.4 CCID class AB, ATR2, Feature2 (also Feature3, Feature4, Feature5)

ICC class AB (3v/5v).

CCID:
- no automatic feature on insertion or for voltage selection.
- class AB (bVoltageSupport=03h, 3v/5v)

---

**Diagram:**

USB HOST -> CHIP CARD INTERFACE DEVICE -> ICC

- **Interrupt - IN**
  - RDR_to_PC_NotifySlotChange
  - bmSlotICCState = xxxx11xxb

- **Cold Reset ICC 3v**

- **ATR2**

- **PC_to_RDR_IccPowerOn**
  - bSlot = 01h, bSeq = 8Bh
  - dwLength = 00000000h
  - bPowerSelect = 02h

- **Bulk - IN**
  - RDR_to_PCM_DataBlock
  - bSlot = 01h, bSeq = 8Bh
  - bStatus=00h with bError =00h
  - abData = ATR2

---

11. APPENDIX H  Samples of CCID diagrams based on dwFeatures field
11.4 Sample diagrams of rate and protocol management

11.4.1 CCID fixed rate (=ATR), ATR 2, Feature1 (or Feature2, Feature3)

ICC:
- only T=0
- TA1 max = 18

CCID:
- Maximum rate = standard ATR rate
- Automatic parameters negotiation with proprietary algorithm, or, Automatic PPS according to current parameters
- Automatic baud rate change according to parameters
- Automatic ICC clock frequency change according to parameters
- Automatic parameters configuration based on ATR data
- APDU or TPDU level

USB HOST

CHP CARD INTERFACE DEVICE

ICC

PC_to_RDR_CARDPOWERON successful on slot 1

ATR2

From ATR data, set parameters to:
- bProtocolNum = 00h
- bmIndexDindex = 11h
- bGuardTimeT0 = 02h
- bWaitingIntegerT0 = 05h

PC_to_RDR_GetParameters
- bSlot = 01h, bSeq = 8Ch

Bulk - IN

RDR_to_PC_Parameters
- bSlot = 01h, bSeq = 8Ch
- bProtocolNum = 00h
- bmIndexDindex = 11h
- bGuardTimeT0 = 02h
- bWaitingIntegerT0 = 05h

Bulk - OUT

PC to RDR CardPowerOn successful on slot 1

CCID ready to exchange APDU or TPDU
11.4.2 CCID with high speed, ATR 4, Feature1 (or Feature2)

ICC:
- T=0 and T=1
- Specific mode with TA1 = 18

CCID:
- Maximum rate > (Maximum frequency * 12) / 372
- Automatic parameters negotiation with proprietary algorithm, or, Automatic PPS according to current parameters
- Automatic baud rate change according to parameters
- Automatic ICC clock frequency change according to parameters
- Automatic parameters configuration based on ATR data
- APDU level

---

**Diagram:**

1. **USB HOST**
2. **CHIP CARD INTERFACE DEVICE**
3. **ICC**

**PC_to_RDR.CardPowerOn successful on slot 1**

From ATR data, set parameters to:
- bProtocolNum = 01h
- bmFindexDindex = 18h
- bmTCCKST1 = 10h
- bGuardTimeT1 = 02h
- bWaitingIntegersT1 = 38h

**Bulk - IN**

**RDR_to_PC_Parameters**
- bSlot = 01h, bSeq = 8Dh
- bProtocolNum = 01h
- bmFindexDindex = 18h
- bmTCCKST1 = 10h
- bGuardTimeT1 = 02h
- bWaitingIntegersT1 = 38h

**Bulk - OUT**

**PC_to_RDR.GetParameters**
- bSlot = 01h, bSeq = 8Dh

**CCID ready to exchange APDU**
11.4.3 CCID fixed rate (=ATR), ATR 2, Feature4 (or Feature5)

ICC:
- only T=0
- TA1 max = 18

CCID:
- Maximum rate = standard ATR rate
- Automatic baud rate change according to parameters
- Automatic ICC clock frequency change according to parameters
- TPDU or Character level
11.4.4 CCID fixed rate (=ATR), ATR 2, Feature6

ICC:
- only T=0
- TA1 max = 18

CCID:
- Maximum rate = standard ATR rate
- No automatic feature
- Character level

---

**USB HOST**

**CHIP CARD INTERFACE DEVICE**

**ICC**

---

**PC_to_RDR_CardPowerOn successful on slot 1**

---

**ATR2**

---

**CCID ready to exchange Characters**

---

11. APPENDIX H  Samples of CCID diagrams based on dwFeatures field
11.4.5 CCID fixed rate (=ATR), ATR 3, Feature1

**ICC:**
- T=0 and T=1
- TA1 max = 18

**CCID:**
- Maximum rate = standard ATR rate
- Automatic parameters negotiation with proprietary algorithm (here for example: use PPS exchange to select maximum rate and to select T=1 when available)
- Automatic baud rate change according to parameters
- Automatic ICC clock frequency change according to parameters
- Automatic parameters configuration based on ATR data
- APDU level

---

**USB HOST**

**CHIP CARD INTERFACE DEVICE**

**ICC**

PC_to_RDR_CardPowerOn successful on slot 1

ATR3

From ATR data, set parameters to:
- bProtocolNum = 01h
- bmFindexDindex = 11h
- bmTCCKST1 = 10h
- bGuardTimeT1 = 02h
- bWaitingIntegersT1 = 38h

PPS Request:
- FF 01 FE

PPS Response:
- FF 01 FE

PC_to_RDR_GetParameters

bSlot = 01h, bSeq = 92h

---

**Bulk-OUT**

**PC_to_RDR_GetParameters**

bSlot = 01h, bSeq = 92h

---

**Bulk - IN**

**RDR_to_PC_Parameters**

bSlot = 01h, bSeq = 92h
bProtocolNum = 01h
bmFindexDindex = 11h
bmTCCKST1 = 10h
bGuardTimeT1 = 02h
bWaitingIntegersT1 = 38h

---

**CCID ready to exchange APDU**

11. APPENDIX H  Samples of CCID diagrams based on dwFeatures field
11.4.6 CCID with high speed, ATR 3, Feature1

ICC:
- T=0 and T=1
- TA1 max = 18

CCID:
- Maximum rate > (Maximum frequency * 12) / 372
- Automatic parameters negotiation with proprietary algorithm (here for example: use PPS exchange to select maximum rate and to select T=1 when available)
- Automatic baud rate change according to parameters
- Automatic ICC clock frequency change according to parameters
- Automatic parameters configuration based on ATR data
- APDU level

---

**Diagram:**
- USB HOST
- CHIP CARD INTERFACE DEVICE
- ICC

**PC_to_RDR_CardPowerOn successful on slot 1**

From ATR data, set parameters to:
- bProtocolNum = 01h
- bmFindexDindex = 18h
- bmTCCKST1 = 10h
- bGuardTimeT1 = 02h
- bWaitingIntegersT1 = 38h

**PPS Request:**
FF 11 18 F6

**PPS Response:**
FF 11 18 F6

**Bulk - OUT**
**PC_to_RDR_GetParameters**
- bSlot = 01h, bSeq = 93h

**Bulk - IN**
**RDR_to_PC_Parameters**
- bSlot = 01h, bSeq = 93h
- bProtocolNum = 01h
- bmFindexDindex = 18h
- bmTCCKST1 = 10h
- bGuardTimeT1 = 02h
- bWaitingIntegersT1 = 38h

**CCID ready to exchange APDU**

---

11. APPENDIX H  Samples of CCID diagrams based on dwFeatures field
11.4.7 CCID with high speed, ATR 3, Feature2 (or Feature3)

ICC:
- T=0 and T=1
- TAI max = 18

CCID:
- Maximum rate > (Maximum frequency * 12) / 372
- Automatic PPS according to current parameters
- Automatic baud rate change according to parameters (host prefers T=1 protocol)
- Automatic ICC clock frequency change according to parameters
- Automatic parameters configuration based on ATR data
- APDU or TPDU level

11. APPENDIX H  Samples of CCID diagrams based on dwFeatures field
11.4.8 CCID with high speed, ATR 3, Feature4

ICC:  - T=0 and T=1
  - TA1 max = 18

CCID:  - Maximum rate > (Maximum frequency * 12) / 372
  - Automatic baud rate change according to parameters (host prefers T=1 protocol)
  - Automatic ICC clock frequency change according to parameters
  - TPDU level

---

**USB HOST**

PC_to_RDR_CardPowerOn successful on slot 1

ATR3

Parameters set to default:
- bProtocolNum = 00h
- bmFindexDindex = 11h
- bGuardTimeT0 = 00h
- bWaitingIntegerT0 = 0Ah

PPS Request:
FF 11 18 F6

PPS Response:
FF 11 18 F6

---

**CHIP CARD INTERFACE DEVICE**

PC_to_RDR_XfrBlock
- bSlot = 01h, bSeq = 96h
- abData = FFh, 11h, 18h, F6h

PC_to_RDR_SetParameters
- bSlot = 01h, bSeq = 97h
- bProtocolNum = 01h
- bmFindexDindex = 18h
- bmTCCKST1 = 10h
- bGuardTimeT1 = 02h
- bWaitingIntegersT1 = 38h

Set parameters to:
- bProtocolNum = 01h
- bmFindexDindex = 18h
- bmTCCKST1 = 10h
- bGuardTimeT1 = 02h
- bWaitingIntegersT1 = 38h

---

**ICC**

PC_to_RDR_Parameters
- bSlot = 01h, bSeq = 96h
- abData = FFh, 11h, 18h, F6h

PC_to_RDR_SetParameters
- bSlot = 01h, bSeq = 97h
- bProtocolNum = 01h
- bmFindexDindex = 18h
- bmTCCKST1 = 10h
- bGuardTimeT1 = 02h
- bWaitingIntegersT1 = 38h

Set parameters to:
- bProtocolNum = 01h
- bmFindexDindex = 18h
- bmTCCKST1 = 10h
- bGuardTimeT1 = 02h
- bWaitingIntegersT1 = 38h

---

CCID ready to exchange TPDU at high speed

---

11. APPENDIX H  Samples of CCID diagrams based on dwFeatures field
11.4.9 CCID with high speed, ATR 3, Feature5

ICC:
- T=0 and T=1
- TA1 max = 18

CCID:
- Maximum rate > (Maximum frequency * 12) / 372
- Automatic baud rate change according to parameters (host prefers T=1 protocol)
- Automatic ICC clock frequency change according to parameters
- Character level

---

**USB HOST**

**CHIP CARD INTERFACE DEVICE**

**ICC**

---

**PC_to_RDR_CardPowerOn successful on slot 1**

ATR3

Parameters set to default:
- bProtocolNum = 00h
- bmFindexDindex = 11h
- bGuardTimeT0 = 00h
- bWaitingIntegerT0 = 0Ah

TPDU = PPS Request: FF 11 18 F6

---

**Beginning PPS Response:** FF 11

**Bulk-OUT**

**PC_to_RDR_XfrBlock**
- bSlot = 01h, bSeq = 99h
- wLevelParameter = 0002h
- abData = FFh, 11h, 18h, F6h

**End of PPS Response:** 18 F6

---

**Bulk-OUT**

**PC_to_RDR_XfrBlock**
- bSlot = 01h, bSeq = 99h
- wLevelParameter = 0002h
- abData is absent

**Bulk-OUT**

**PC_to_RDR_SetParameters**
- bSlot = 01h, bSeq = 9Ah
- bProtocolNum = 01h
- bmFindexDindex = 18h
- bmTCCKST1 = 10h
- bGuardTimeT1 = 02h
- bWaitingIntegerT1 = 38h

Set parameters to:
- bProtocolNum = 01h
- bmFindexDindex = 18h
- bmTCCKST1 = 10h
- bGuardTimeT1 = 02h
- bWaitingIntegerT1 = 38h

**Bulk-OUT**

**RDR_to_PC_Parameters**
- bSlot = 01h, bSeq = 9Ah
- bProtocolNum = 01h
- bmFindexDindex = 18h
- bmTCCKST1 = 10h
- bGuardTimeT1 = 02h
- bWaitingIntegerT1 = 38h

C CID ready to exchange characters at high speed
11.4.10 CCID with high speed, ATR 3, Feature6

**ICC:**
- T=0 and T=1
- TA1 max = 18

**CCID**
- Maximum rate > (Maximum frequency * 12) / 372
- No automatism
- Character level

---

**USB HOST**

**CHIP CARD INTERFACE DEVICE**

**ICC**

---

PC_to_RDR_CardPowerOn successful on slot 1

ATR3

Parameters set to default:
- bProtocolNum = 00h
- bFinfexDIndex = 11h
- bGuardTimeT0 = 00h
- bWaitingIntegerT0 = 0Ah

TPDU = PPS Request:
FF 11 18 F6

Beginning PPS Response: FF 11

---

PC_to_RDR_XfrBlock
bSlot = 01h, bSeq = 9Bh
wLevelParameter = 0002h
abData = FFh, 11h, 18h, F6h

Bulk - OUT

---

PC_to_RDR_XfrBlock
bSlot = 01h, bSeq = 9Ch
abData is absent

Bulk - OUT

---

PC_to_RDR_Parameters
bSlot = 01h, bSeq = 9Dh
bProtocolNum = 01h
bFinfexDIndex = 18h
bTCCST1 = 10h
bGuardTimeT1 = 02h
bWaitingIntegerT1 = 38h

Set parameters to:
- bProtocolNum = 01h
- bFinfexDIndex = 18h
- bTCCST1 = 10h
- bGuardTimeT1 = 02h
- bWaitingIntegerT1 = 38h

Bulk - IN

---

RDR_to_PC_Parameters
bSlot = 01h, bSeq = 9Dh
bProtocolNum = 01h
bFinfexDIndex = 18h
bTCCST1 = 10h
bGuardTimeT1 = 02h
bWaitingIntegerT1 = 38h

Set parameters to:
- bProtocolNum = 01h
- bFinfexDIndex = 18h
- bTCCST1 = 10h
- bGuardTimeT1 = 02h
- bWaitingIntegerT1 = 38h

Bulk - IN

---

PC_to_RDR_XfrBlock
bSlot = 01h, bSeq = 9Ch
wLevelParameter = 0002h
abData = FFh, 11h, 18h, F6h

Bulk - OUT

---

RDR_to_PC_Parameters
bSlot = 01h, bSeq = 9Dh
bProtocolNum = 01h
bFinfexDIndex = 18h
bTCCST1 = 10h
bGuardTimeT1 = 02h
bWaitingIntegerT1 = 38h

Set parameters to:
- bProtocolNum = 01h
- bFinfexDIndex = 18h
- bTCCST1 = 10h
- bGuardTimeT1 = 02h
- bWaitingIntegerT1 = 38h

Bulk - IN

---

RDR_to_PC_DataBlock
bSlot = 01h, bSeq = 9Ch
abData = FFh, 11h, 18h, F6h

Bulk - OUT

---

RDR_to_PC_DataBlock
bSlot = 01h, bSeq = 9Ch
abData = FFh, 11h, 18h, F6h

Bulk - OUT

---

Continuation on next page

11. APPENDIX H  Samples of CCID diagrams based on dwFeatures field
11. APPENDIX H  Samples of CCID diagrams based on dwFeatures field
11.4.11 CCID high speed, “EMV like”, Cold ATR : ATR1, Warm ATR : ATR4, Feature1

ICC :
- class AB (5v/3v)
- Cold ATR : negotiable mode, only T=0, TA1 max=18
- Warm ATR : specific mode, T=1, TA1=18

CCID
- Automatic ICC activation on insertion
- Automatic voltage selection
- class A
- Maximum rate > (Maximum frequency * 12) / 372
- Automatic parameters negotiation with proprietary algorithm (here “EMV like”: try warm reset when ATR rejected because not EMV)
- Automatic baud rate change according to parameters
- Automatic ICC clock frequency change according to parameters
- Automatic parameters configuration based on ATR data
- APDU level

---

USB HOST

CHIP CARD INTERFACE DEVICE

ICC

Icc Inserted

<table>
<thead>
<tr>
<th>Cold reset ICC 5V</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATR rejected: Warm reset ICC 5V</td>
</tr>
</tbody>
</table>

From ATR data, set parameters to :
- bProtocolNum = 01h
- bmTCCKST1 = 10h
- bGuardTimeT1 = 02h
- bWaitingIntegersT1 = 38h

Interrupt - IN

RDR_to_PC_NotifySlotChange
- bSlotICCState = xxxx11xx

PC_to_RDR_IccPowerOn
- bSlot = 01h, bSeq = A0h
- dwLength=00000000h
- bPowerSelect=01h

PC_to_RDR_Parameters
- bSlot = 01h, bSeq = A1h
- dwLength – length of abData field
- abData = ATR4

Bulk - IN

RDR_to_PC_DataBlock
- bSlot = 01h, bSeq = A0h
- bStatus=00h with bError =00h
- dwLength – length of abData field
- abData = ATR4

Bulk - IN

RDR_to_PC_Parameters
- bSlot = 01h, bSeq = A1h
- bProtocolNum = 01h
- bmTIndexDIndex = 18h
- bGuardTimeT1 = 02h
- bWaitingIntegersT1 = 38h

---

11. APPENDIX H  Samples of CCID diagrams based on dwFeatures field

Page 132 of 135  20 Mar 01 @ 8:53
11.5 Sample diagrams of automatic IFSD management

11.5.1 CCID with large IFSD, ATR4, Feature1 (or Feature2)

ICC:
- only T=1
- TA1 max = 18

CCID:
- dwMaxIFSD = 254
- Slot NAD value previously set to 12h
- Automatic IFSD exchange
- APDU level

```
<table>
<thead>
<tr>
<th>USB HOST</th>
<th>CHIP CARD INTERFACE DEVICE</th>
<th>ICC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**Parameters are:**
- bProtocolNum = 01h
- bmIndexDindex = 18h
- bTCKST1 = 10h
- bGuardTimeT1 = 02h
- bWaitingIntegersT1 = 38h
- bNadValue = 12h

**S(IFSD Request):**
12 C1 01 FE 2C

**S(IFSD Response):**
21 E1 01 FE 3F

**T=1 exchanges APDU transportation**

```
<table>
<thead>
<tr>
<th>PC_to_RDR_XfrBlock</th>
</tr>
</thead>
<tbody>
<tr>
<td>bSlot = 01h, bSeq = A2h</td>
</tr>
<tr>
<td>bLevelParameter = 0000h</td>
</tr>
<tr>
<td>abData = &lt;first APDU command of the session&gt;</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>RDR_to_PC_DataBlock</th>
</tr>
</thead>
<tbody>
<tr>
<td>bSlot = 01h, bSeq = A2h</td>
</tr>
<tr>
<td>bChainParameter = 00h</td>
</tr>
<tr>
<td>abData = &lt;first APDU response of the session&gt;</td>
</tr>
</tbody>
</table>
```
11.5.2 CCID with large IFSD, ATR4, Feature4

ICC:
- only T=1
- TA1 max = 18

CCID:
- dwMaxIFSD = 254
- Slot NAD value previously set to 12h
- Automatic IFSD exchange
- TPDU level

11. APPENDIX H  Samples of CCID diagrams based on dwFeatures field
## 12. APPENDIX I  Error Codes

Slot Error register:

<table>
<thead>
<tr>
<th>Error Name</th>
<th>Error Code</th>
<th>Possible Causes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMD_ABORTED</td>
<td>-1 (FFh)</td>
<td>Host aborted the current activity</td>
</tr>
<tr>
<td>ICC_MUTE</td>
<td>-2 (FEh)</td>
<td>CCID timed out while talking to the ICC</td>
</tr>
<tr>
<td>XFR_PARITY_ERROR</td>
<td>-3 (FDh)</td>
<td>Parity error while talking to the ICC</td>
</tr>
<tr>
<td>XFR_OVERRUN</td>
<td>-4 (FCh)</td>
<td>Overrun error while talking to the ICC</td>
</tr>
<tr>
<td>HW_ERROR</td>
<td>-5 (FBh)</td>
<td>An all inclusive hardware error occurred</td>
</tr>
<tr>
<td>BAD_ATR_TS</td>
<td>-8 (F8h)</td>
<td></td>
</tr>
<tr>
<td>BAD_ATR_TCK</td>
<td>-9 (F7h)</td>
<td></td>
</tr>
<tr>
<td>ICC_PROTOCOL_NOT_SUPPORTED</td>
<td>-10 (F6h)</td>
<td></td>
</tr>
<tr>
<td>ICC_CLASS_NOT_SUPPORTED</td>
<td>-11 (F5h)</td>
<td></td>
</tr>
<tr>
<td>PROCEDURE_BYTE_CONFLICT</td>
<td>-12 (F4h)</td>
<td></td>
</tr>
<tr>
<td>DEACTIVATED_PROTOCOL</td>
<td>-13 (F3h)</td>
<td></td>
</tr>
<tr>
<td>BUSY_WITH_AUTO_SEQUENCE</td>
<td>-14 (F2h)</td>
<td>Automatic Sequence Ongoing</td>
</tr>
<tr>
<td>PIN_TIMEOUT</td>
<td>-16 (F0h)</td>
<td></td>
</tr>
<tr>
<td>PIN_CANCELLED</td>
<td>-17 (EFh)</td>
<td></td>
</tr>
<tr>
<td>CMD_SLOT_BUSY</td>
<td>-32 (E0h)</td>
<td>A second command was sent to a slot which was already processing a command.</td>
</tr>
</tbody>
</table>

| User Defined                     | -64 to –127|                                        |
|                                  | (C0h-81h)  |                                        |
| Reserved for future use          |            | All others ( 80h and those filling the gaps) |

---

**AA12. APPENDIX I  Error Codes**  
Page 135 of 135  
20 Mar 01 @ 8:53