

by Mats Hansson, Intertex Data AB
Copyright ©Intertex Data AB

Smartcard reader in external TCP/IP device (integrated into a Firewall, ADSL modem). Protocol between host and reader.

1. General

This document describes the specific Intertex protocol for smartcard (IC card) reader integrated into a firewall and/or an ADSL modem, and using TCP/IP to communicate with the host computer. The information is intended mainly for developers of drivers and API:s. For application programmers on the Windows platform, the related document describing the Application Program Interface is more relevant, though the information below may be of some use.

The protocol assumes error-free connections. This is accomplished by the TCP protocol.

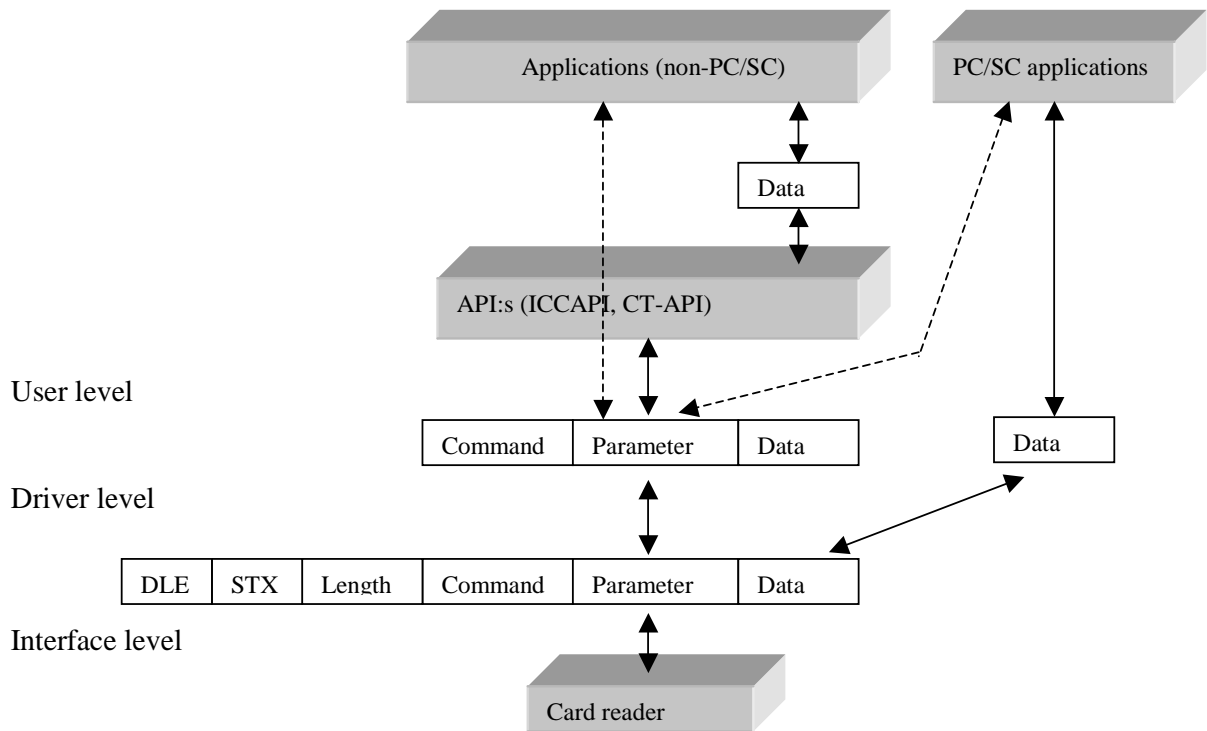
There are two levels of this protocol:

- user level
- driver level

The user level only bothers about the command byte, the parameter byte and the data field. Of these fields, the normal application is mostly dealing with the data field only. This is typically the card data entered/received when using the Intertex API *dll*-functions (such as `ICC_Data_To_Card`) and the related, top-level API:s CT-API and PC/SC. However, the full user level protocol, giving access to the full repertoire of reader commands, may be accessed through “direct” modes. There are functions for “direct” mode in all the API:s mentioned.

The user level API:s knows the size of the message (command + parameter + data) and gets the size of the response, all through normal C-language parameters.

At the next lower level, handled by the drivers or by the *dll*-libraries, a small header is added preceding the command byte. The header contains a start tag (DLE and STX characters) followed by the length of the user level data. This is the packet that is transferred by the TCP protocol to the reader.



The dashed lines imply the “direct” modes described above.

The data to/from the card are always direct conversion: for card with inverse convention (see ISO 7816-3) the inversion of characters is being taken care of by the reader, both when writing to the card and when reading from the card. The host does not have to bother about the inverse convention at all.

2. Format of messages/responses at the user level

Message format, from host to reader:

(unless otherwise stated, all numbers are decimal)

<COMMAND>	<PARAMETER>	<DATA>
1 byte According to list below	1 byte Command-dependent	variable (may not be present)

Summary of commands:

Command	Name	Parameter	Data	Expected response
0	Set V _{pp} on/off	0/1	-	Done
1	Get entire ATR (answer to reset)	not used (=0)	-	ATR string
2	Deactivate	not used (=0)	-	Done
3	Get status	not used (=0)	-	Card status
4	Test card	not used (=0)	-	Card type
6	Get timeout time	not used (=0)	-	Timeout time in 0.1 seconds
9	Get configuration	not used (=0)	-	Configuration bitmask
20	Activate asynchronous card	T value	-	Historical bytes
21	Data to asynchronous card	0	x	Card response
22	Data from asynchronous card	not used (=0)	x	Card response
23	Change frequency	Freq.*2	-	Done
24	Get specific interface parameters	not used (=0)	-	Protocol dependent data
25	Activate with any protocol	not used	-	Protocol and historical bytes
30	Activate synchronous card	Card type	-	Card dependent
31	Data to synchronous card	0	x	Card response
32	Data from synchronous card	not used (=0)	x	Card response
60	Show "Crd" on display	Time (1/10 s)	-	Done
61	Toggle "Crd"/"In" on display	Time (1/10 s)	-	Done
62	Toggle "Crd"/"Out" on display	Time (1/10 s)	-	Done
63	Flash "Err" on display	Time (1/10 s)	-	Done
64	Flash "Pin" on display	Time (1/10 s)	-	Done
65	Restore display	not used (=0)	-	Done
66	Beep (on reader speaker)	Time (1/10 s)	-	Done
70	Reserved	New state	-	State has changed

For a description of the commands and responses, see later in this document.

Response format, from reader to host:

<COMMAND>	<PARAMETER>	<DATA>
1 byte Echo of command from host	1 byte Command dependent If >127: Error code	Variable (may not be present)

Typical codes in parameter byte:

Code	Name	Comments
126	Done, OK	May be replaced by command specific data
128	Card removed	(unexpectedly) If this code has been sent by the reader, as response to any command, the card has been deactivated. The card may then not longer be present, or the same or another card has been reinstalled. In these cases a new activate command has to be issued.
129	Unresponsive card	Time outs etc.
130	Parity errors	At least three errors during a card access
131	Wrong card type	Card type does not match expected
132	Unknown card type	No support of card type
133	Illegal command	Illegal command or command not implemented
134	Card not activated	Card must be activated before requested operation
135	Unnormal SW1SW2	SW1, SW2 was not 9000 _{HEX} (for T=0)
136	Illegal parameter	
137	No support of V _{pp}	Desired programming voltage not available. (At present only 5V is supported)
138	ATR corrupted	Answer to reset string corrupted
139	Error in PTS procedure	The protocol type selection procedure went wrong
140	Bad procedure byte	Was neither ACK, NULL nor SW1 (for T=0)
141	SW1SW2 too early	SW1SW2 was received before all data was transferred (T=0)
142	Illegal command format	Could be frame error (<DLE><STX> missing etc.).
143	Card is T=0	Other T value than 0 was expected
144	Card is T=1	Other T value than 1 was expected
145	Illegal value of TS byte	TS is first byte of Answer-to-reset string
146	Bad length	Length of message is wrong
147	No data available	

3. The driver level protocol

The driver introduce very little overhead on the user protocol when sending and receiving to/from the card reader. For the reader accessible by TCP/IP, the TCP protocol and lower protocol levels take care of all error checking, physical transmission etc.

The driver level protocol consists of four bytes preceeding the command byte:

<TAG>	<LENGTH>	<COMMAND><PARAMETER><DATA >
2 bytes DLE,STX (hex: 0x10,0x02) Used for recognition only.	2 bytes Low byte first. Value is total length, tag and length fields are included.	Variable length. The user level protocol.

The format is the same in both send and receive directions.

A buffer size of 275 will be enough for carrying this protocol, since card data is limited by ISO 7816-3 definition.

The reader is accessed by the computer (driver, API dll-file) by opening a TCP connection on port 5320. The TCP connection shall be opened “just-in-time” when an application needs to access the smartcard reader, and kept open until that application no longer wants the card reader or an inserted smartcard. After that, the connection should be closed. During the open time, the host PC (driver) may send driver level commands just described, and may also check for any received (spontaneous) “new state” messages.

If the reader is powered off and on, the connection will be lost “ungracefully” and the driver/API will not be notified. Thus, the driver/API must have proper error checking and timeouts on all card reader accesses, so it can signal to the application to reconnect (or even try reconnect itself).

Alternatively to port 5320, the reader may be configured to answer on port 6779 instead (coming feature), so any driver/API should be written to try the 2:nd port number if the 1:st one fails. The port 6779 is for (possibly) shared usage of a reader, with interlocking, denying all access from other hosts on the local network while it has been opened. So a driver/API along with its client applications should take care not to keep the port opened for long times. If a driver/API is designed to keep the port opened from startup of the host computer (like the Windows PC/SC), the port 6779 shall not be tried at all.

4. Command description

Command 0: Set programming voltage V_{pp} on or off

This command is used for changing state of V_{pp} . However, for asynchronous cards (T=0 and T=1) this command should normally not be used, since the card itself prompts for V_{pp} on/off. The card reader acts upon these prompts (most cards do not have a separate V_{pp}).

When used, the command contains the desired state in the parameter:

Parameter value	
1	Set V_{pp} in active state
0	Set V_{pp} in idle state.

Response may be:

Parameter value	
126	Done
137	An activation of V_{pp} was requested but the card reader could not support the proper voltage as specified by the card in the Answer To Reset string.

Command 1: Get entire ATR

The full Answer-to-Reset string is returned, including historical bytes. A response for a T=0 card is typically:

<1><126><ATR-response>

Specific error codes might be 128,129, 130, 138 and 145.

ATR string is sent in all cases except in case of error codes 128 or 129.

The command does not change the inactive/active state. If the card was inactive, it activates the card temporarily to get the ATR string, then deactivates it again. The command tries to recover any ATR string regardless of card type.

Command 2: Deactivate

Switch off power supply to card, marks inactive state internally in IC card task of reader.

Response on this command is most probably "Done, OK", even if card was already inactive or even absent.

Command 3: Get status

Used for reporting basic status of the card/card reader.

Response may be:

Parameter value	
1	Card present (properly inserted) but not activated
2	Card activated
3	Card absent and has not been inserted since last "get status" command.
4	Card absent but was inserted since last "get status" command.

Note: In this case, "card removed" is *not* reported as error code 128, since it may be a quite expected response to the query.

Command 4: Test card

Used for getting information about type of card.

Response may be:

Parameter value	
0 - 15	T-value of an asynchronous card
16 - 31	T-value plus 16, indicating the primary T-value but also that the card supports more than one protocol. The full protocol support has to be checked by the host through the "get entire ATR"-command. (A possible strategy is also to try to activate, specifying the desired T-value)
32 - 47	Synchronous card type (not supported yet).

Specific error codes are 128, 129, 132 and 138.

Note: Though actually accessing the card by this command, the card has to be *activated* before any read/write operations.

Command 6: Get timeout time

This command is used to get a suggestion for a suitable timeout time, i.e. the maximum time for the host to wait for an answer from the card reader. The maximum time may depend on card type and clock frequency used in the card reader. It is calculated based upon a 256 byte data transfer to or from the card. In all cases, the card reader will check for timeouts of the card according to specifications in ISO 7816-3 (block and character waiting times) and report such timeouts to the host (by error code 129). Thus, the host should not overdo the timeout checking and could use a rather long timeout time.

The timeout time reported does not include the cryptographic calculations which may be substantially longer than all the other card operations. The host should add at least 5 seconds to the timeout time when such a calculation has been ordered.

The default timeout time is 5 seconds (value=50).

The command may be sent when a card has been activated. If not activated, the reader will always suggest the default value.

The response is <6><Time in 0.1 second units>.

Command 9: Get configuration

The command is issued to get information about the IC-card readers hardware and software configuration (capabilities). The response contains a bit mask in its parameter byte, and one data byte containing a product code:

<9><126><Bit mask><Product code byte>

The product code can be any number 0-255 and identifies the product/brand name.

The bit mask is as follows:

The card reader has:

Bit 0	Three digit LED display
Bit 1	Numerical keypad (keys 0-9, *,#,↑,↓,∞,R)
Bit 2	Beep tone
Bit 3	Alphanumeric keyboard / multilined screen
Bit 4-6	Reserved for future use
Bit 7	Always null (otherwise the parameter byte is an error code)

For compatibility purposes, whenever the bit 3 is set, the bit 0 and 1 is also set.

Command 20: Activate asynchronous card

Sent by the host to activate the card and to get the historical data. The expected protocol T value is sent as the parameter. If the card does not support the protocol, error code 131, 143 or 144 is received from the reader and the card is not activated. In case of 131, the commands "test card" or "get entire ATR" can be used for detecting mysterious cards. If already activated, no actual reset is performed. In this case, if a true reset is wanted, the host must first deactivate the card (by command no. 2).

Expected response:

<20><126><historical bytes>

Specific error codes are 128,129, 131, 137, 143 or 144. For these error codes, no historical bytes are transmitted.

Command 21: Data to asynchronous card

The normal command for conveying data to the IC card, whether it is file selection, keys, PIN-codes or other file contents. The card must have been previously activated.

The format is:

<21><0><data (for T=0: typically 5-byte header + data)>

Expected response is:

<21><126 or error code><card response>

T=0

For T=0, a successful response is:

<21><126><SW1SW2>

SW1SW2 is 90_{HEX}00_{HEX} if no errors. Other values are warned by error code 135 or 141 but may not necessarily indicate an error (i.e. GET RESPONSE handling for EMV cards).

The procedure byte is not sent to the host (unless it was SW1).

Responses in case of error take two forms:

<21><xx><SW1,SW2>

for error codes xx=135 and 141.

<21><xx>

for error codes xx= 128, 129, 130, 134, 137, 140 and 146.

T=1

The T=1 protocol is handled transparently by the IC card reader. It relays any block sent from the host directly to the IC card, after stripping off the host-to-reader header. Likewise, any answer by the card is sent back to the host, framed by the reader-to-host header. As a consequence, all the scenarios described in ISO 7816-3/Amd.1 should be handled by the host. This includes CRC or LRC checking of the prologue and information field of the T=1 block, block chaining, request for change of information field size (IFSC and IFSD) etc. This goes for the request for waiting time extension too, in this case the card reader monitors the S(WTX response) as it has to allow for a longer timeout and not report "unresponsive card" to the host. Typically, as in the CT-API and PC/SC program interfaces, all this protocol handling is taken care of by intermediary drivers/dll:s and need not be dealt with in the application.

The command 24 might be helpful in setting up some parameters for the T=1 driver software in the host.

The format is typically:

<21><0><prologue + information + epilogue fields>

Expected response is:

<21><126><prologue + information + epilogue fields>

or, if an error occurred:

<21><xx>

for error codes xx= 128, 129, 130, 134, 137 and 146.

Command 22: Data from asynchronous card

The normal read command. The card must have been previously activated.

The format is:

<22><0><data (For T=0: typically 5-byte header)>

Expected response is:

<22><126 or error code><card response>

T=0

For T=0, a successful response is:

<21><126><card data + SW1SW2>

SW1SW2 is 90_{HEX}00_{HEX} if no errors. Other values are warned by error code 135 or 141 but may not necessarily indicate an error (i.e. GET RESPONSE handling for EMV cards).

The procedure byte is not sent to the host (unless it was SW1).

Responses in case of error take three forms:

<21><xx><data + SW1,SW2>

for error code xx=135.

<21><xx><SW1,SW2>

for error code xx=141.

<21><xx>

for error codes xx= 128, 129, 130, 134, 137 and 140.

Note: For commands to the card that does not lead to any data being transferred, the command "data to" (no. 21) should be used rather than the "data from" (no. 22) command. When P3 in

the header is null in a "data from" command, the card reader expects 256 byte to be transmitted from the card (see ISO 7816-3, clause 8.3.2).

T=1

As the block structure for T=1 is independent of the data direction, the internal procedures in the IC card reader are the same whether data is transferred to or from the card. Thus, the host can use any of commands 21 and 22 for all T=1 transmissions and does not have to separate between "data from" and "data to" commands. (See command 21 for description).

Command 23: Change frequency

For asynchronous cards, the reader chooses a card frequency according to the contents of the ATR-string from the card. The chosen frequency is as high as allowed according to the recommendations in ISO 7816-3, table 6. However, with detailed knowledge about type and brand of the inserted card, the host may switch to another frequency. For example, Philips DX cards may be run on 8 Mhz which is faster than the recommendations according to its clock rate conversion factor (F) in the ATR string. The switch to the higher frequency is performed by command 23, and the IC card reader adjusts waiting times and bit rates (to the card) accordingly.

The format is:

<23><x>

where x is frequency in MHz multiplied by 2. If, for example, 6.5 Mhz is wanted, x should be set to 13.

Specific error codes are 128, 134 and 136. The chosen frequency is not checked against any limits at all.

Command 24: Get specific interface parameters

With this command, some interface parameters can be retrieved from the ATR string. The parameters are specific to the protocol in use. The parameters may also be retrieved by the "get entire ATR" command, but the command 24 relieves the host from some calculations (of the BWT/WWT) and from dealing with the structure of the ATR string.

The parameters are:

T=0

WWT	Work waiting time, in 0.1 second units. It covers, with margins, the maximum time between a character sent by the card and the previous character, sent by the card or by the IC card reader. The WWT value is calculated with regard to the card frequency in use.
-----	---

T=1

IFSC	Information field size for the card. Value is 1-254 (32 is default). The value is taken from the ATR string and is not changed by any IFS request block from the host.
BWT	Block waiting time, in 0.1 second units. It covers, with margins, the maximum time from the last character received by the card to

the first character being sent from the card. If used by the host in timeouts, at least 0.5 s should be added to allow for both card-to-reader and reader-to-host transmissions. The BWT value is calculated with regard to the card frequency in use.

EDC type Form of error detection code to be used in the epilogue field of the block frame for T=1. Value=0 is LRC, value=1 is CRC.

The character waiting time (CWT) and the guardtimes (BGT and N) are dealt with in the reader only. The IFSD (information field size for the interface device) is 32 by default and can be changed to a higher value as described in ISO 7816-3, Amd. 1, by the use of an IFS request block. The reader can support an IFSD of max. 254 bytes.

The format of the response is:

For T=0: <24><126><WWT>

and for T=1: <24><126><IFSC><BWT><EDC type>

Specific error codes are 128, 134 and 147. The data bytes are not transmitted after these codes.

Command 25: Activate with any protocol

As command 20 but does not make any assumptions which protocol is selected by the user. If the card supports at least T=0, that protocol is selected. If the card supports T=1 but not T=0, the T=1 is selected. As command 20, the historical bytes are returned, preceded by the protocol number selected by the reader.

If already activated, no actual reset is performed, the protocol and the historical bytes returned as expected. In this case, if a true reset is wanted, the host must first deactivate the card (by command no. 2).

Expected response:

<20><126><protocol number><historical bytes>

The protocol number returned is one byte, 0 stands for T=0 etc.

Specific error codes are 128,129, 131 or 137. For these error codes, no historical bytes nor protocol number are transmitted.

Command 30-32: Synchronous card operations

Not implemented yet. Message structure and behaviour will depend on card type as specified in parameter of command 30 (activate).

Command 60: Show "Crd" on display

Displays the text "Crd" on the reader for a specified amount of time. The time in 0.1 second units is specified in the parameter. A value 0 means "forever" (until other display command). The response is <60><126>.

Command 61: Toggle "Crd"/"In" on display

Makes the reader display toggle between the text "Crd" and the text "In" with a specified frequency. The time (for each text) in 0.1 second units is specified in the parameter. A value 0 means a stable text "In" (until other display command). The response is <61><126>.

Command 62: Toggle "Crd"/"Out" on display

As above, but text "Out" instead of "In".

Command 63: Flash "Err" on display

As above, but toggling between "Err" and a blank display. A value 0 means a stable text "Err".

Command 64: Flash "Pin" on display

As for "Err" but displaying the text "Pin".

Command 65: Restore display

Restore the display to normal behaviour. The response is <65><126>.

ATT: The commands 60-64 may conflict with the readers own use of the display. The behaviour will depend on what is currently displayed, toggling times etc. but will generally be acceptable.

Command 66: Beep

(Kept for compatibility, the reader in the firewall/ADSL modem currently has got no beeper). Makes the reader speaker beep for a specified length of time. The time value is given in the parameter in 0.1 second units. Max value is 30 (3 seconds). If bit 7 is set in the parameter (128 is added), the beep will use a higher frequency. The value 132 thus gives a high frequency beep of 0.4 seconds. The response is <65><126>.

The beep command is only allowed when the reader has got such a feature (see command no. 9), otherwise the error 133 is returned.

Command 70: New status

This is really not a command but is a message sent spontaneously from the TCP/IP based smartcard reader when a card is inserted or removed. The new status (coded like the "Get status" command) follows as the parameter. A driver/API should be designed to always expect "new status" messages in the TCP packet received. To decrease the network overhead, the driver/API could check for such incoming messages on a regular basis, rather than polling the reader by the "get status" command (command no. 3). In many cases, it may be sufficient to check the state when needed, i.e. before a card read/write operation.