CT - A P I 1.1

Application Independent
Card**T**erminal **A**pplication **P**rogramming **I**nterface
for ICC Applications

Authors:
Deutsche Telekom AG / PZ Telesec
SIT – Fraunhofer Institut für Sichere Telekooperation
TÜV Informationstechnik GmbH
T*ELE****T****RUS****T*** Deutschland e.V.
Date: 14.10.98

Version 1.0

15.04.2002
updated from 15.10.1998

Editor: Jürgen Atrott, ***TÜV*** Informationstechnik GmbH

Contents

Preface ... 3
1. Scope ... 3
2. Normative references ... 3
3. Abbreviations ... 3
4. CT-API ... 3
4.1 CT-API structure ... 3
4.2 Type definition ... 4
4.3 CT-API functions ... 4
4.3.1 CT_init ... 4
4.3.2 CT_data ... 5
4.3.3 CT_close ... 6
4.4 General function values of the CT-API functions ... 6
Annex A (normative) ... 7
Identification of CT API Functions with a 'well known identifier' ... 7
Identification of CT API file names with a 'well known identifier' ... 7
Annex B (informative) ... 8
Information for Programmers ... 8
Windows 16 Bit DLL ... 8
Annex C (informative) ... 9
Programming Example ... 9
Annex D (informative) ... 10
Example of an internal processing of the CT-API function CT_data ... 10

Authors (contacts):

Deutsche Telekom AG/ PZ Telesec
Untere Industriestr.20
57250 Netphen

R. Moos
Tel.: (0271) 708-1600, Fax: -1625
E-mail: Rainer.Moos@telekom.de
http://www.telesec.de

Fraunhofer Institut für Sichere Telekooperation
Rheinstrasse 75
64295 Darmstadt

L. Eckstein, B. Struif
Tel.: (06151) 869-205, Fax: -224
E-mail: Levona.Eckstein@sit.fraunhofer.de
http://www.sit.fraunhofer.de

TÜV Informationstechnik GmbH
Leimbachstr. 227
D 57074 Siegen

J. Atrott
Tel.: (0271) 3378-194, Fax: -197
E-mail: J.Atrott@tuvit.de
http://www.tuvit.de

TELETRUST Deutschland e.V. Geschäftsstelle
Chamissostraße 11
99096 Erfurt

Prof. Dr.H. Reimer
Tel.: (0361) 3460531, Fax: (0361) 3453957
E-mail: teletrust@t-online.de
http://www.teletrust.de

Other addresses:

RID German National Registration Authority
c/o Fraunhofer Institut Sichere Telekooperation
Rheinstr. 75
64295 Darmstadt

B. Struif
Tel.: (06151) 869-206, Fax: -224
E-mail: Bruno.Struif@sit.fraunhofer.de

Preface

This version CT-API 1.1 is functionally compatible with CT-API dated 29.10.93.
The following changes were made:

- *Amendment / change of dad / sad addresses*
- *Addition of a type definition*
- *Change of ctn and pn parameters to 16 bit format*
- *Change of lenr and response parameters in CT_data*
- *Extension of error return codes*
- *Addition of a well known identifier for simultaneous use of different CT-API libraries and different CT-API files of several manufacturers*
- *Revision of texts and diagrams*
- *Additional authors*

1. Scope

This specification describes the application independent CardTerminal functions that are required to simplify the handling of and communication with IC cards.
The CT-API functions are constructed so that

- memory cards and processor cards can be used
- the necessary commands to control the CardTerminal can be transmitted and
- CardTerminals can be addressed independent of the port.

At the ***CT-AP Interface*** it is not visible whether the CardTerminal is a fitted model or an external unit.

A CT-API function library is a software component of the CardTerminal and should be provided by the CT manufacturer for those system environments that are considered relevant.

2. Normative references

ISO 3166: 1994 Codes for the representation of names of countries

ISO/IEC 7816-3: 1989/1993 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 - Inter-industry commands for interchange

3. Abbreviations

API	Application Programming Interface
CT	Card Terminal
ctn	Card Terminal Number
CTM	Card Terminal Manufacturer
dad	Destination ADdress
ICC	Integrated Circuit(s) Card
HTSI	Host Transport Service Interface
lenc	LENgth Command
lenr	LENgth Response
OS	Operating System
pn	Port Number
res	RESult of function
RFU	Reserved for Future Use
RID	Registered application provider identifier
sad	Source ADdress
wki	Well Known Identifier

4. CT-API

4.1 CT-API structure

The CT-API functions are provided by a so-called HTSI module (Host-TransportService-Interface). The embedding of an HTSI module in its system environment is shown in fig.1.

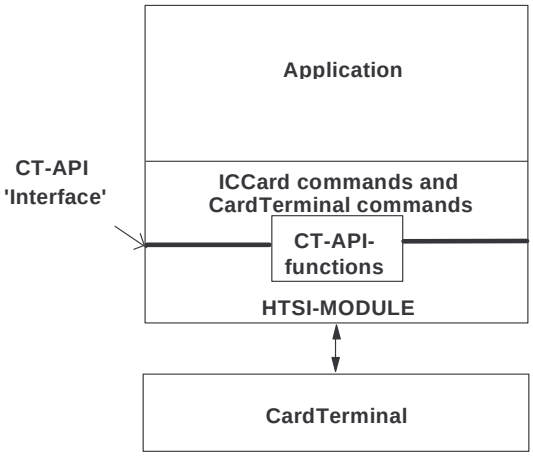

Fig.1 The embedding of an HTSI Module in its system environment

4.2 Type definition

The CT API uses functions whose parameters and function results are described in the following notation:

Ordinal Type	Meaning
IUx	Integer unsigned Index x shows the used bit length in decimal form (i.e.: IU16 - 0 ... 65535)
ISx	Integer signed Index x shows the used bit length in decimal form (i.e.: IS8 =-128 ... 127)

Tab. 1: Type notation

4.3 CT-API functions

The CT-API contains the following functions which are provided by a HTSI module.

CT-API-Function	Meaning
CT_init	Initiation of the host/CT connection
CT_data	Transmission of a command to a CardTerminal or to an ICC and give back the response
CT_close	Close the host/CT connection

Tab.2: CT-API functions

When a function is properly executed the function value 0 is returned, otherwise a negative function value shows the cause of the error. The function values for all CT-API functions are listed in section 4.4.

Normative and informative details about the use of CT-API functions are given in Annexes B and C.

In order to ensure compatibility CT-API functions shall have the same name and parameters in every implementation.

Additionally in the case of simultaneous use of several CT-API libraries in one application system a CT-API library with 'well known identifier' shall be provided (see annex A).

4.3.1 CT_init

With the function CT_init the host/CT interface is selected (e.g. COM1: or COM2: for MS-DOS and e.g. ttya or ttyb for UNIX) and initiated. The default values for the communication are herewith automatically set. The CT_init function must be called before communication begins with the CardTerminal and the ICC. At the same time the CardTerminal is assigned to a unique logical address in the form of a CardTerminal Number (ctn) which may be freely chosen by the programmer. Also a port number (pn) is used. It denotes the physical interface via which the CardTerminal is accessed.
The port number is not regulated by this specification and may be chosen according to the manufacturer's description.
The assignment or the assigning principle of the port number to the physical interface can be seen from the manufacturer's documentation.
One CardTerminal manufacturer for example may assign the port number 1 to a keyboard interface, while another manufacturer may use port number 1 for the outlet 1 of an 8 bit wide I/O or for the serial interface COM1: of a PC.
The application program can nevertheless address these different port models with CardTerminal number 1 and port number 1 as long as the manufacturer's API library is linked. Thus the modules concerned in the application program do not need to be modified.

Function:

CT_init (ctn, pn)

Function parameters:

Parameter name	Parameter type	Meaning
ctn	Input-parameter, type IU16	Logical CardTerminal number
pn	Input-parameter, type IU16	Port number of the physical interface

Tab 3: Parameters for CT_init

Function values:

Parameter type	Meaning
Output parameter, Typ IS_8	Function values see section 4.4

Tab 4: Function values of CT_init

4.3.2 CT_data

The function CT_data is used to send an ICC command to a card or a CardTerminal command to an integrated or connected CardTerminal and returns the command response to the calling program.

Function:

CT_data (ctn, dad, sad, lenc, command, lenr, response)

Function parameters:

Parameter name	Parameter type	Meaning
ctn	input-parameter, type IU16	Logical CardTerminal number
dad	Input / Output-parameter, type IU8	Destination address
sad	I/O-parameter, type IU8	Source address
lenc	Input-parameter, type IU16	Command length in byte
command	Input-parameter Reference address of a field with elements of type IU8 which contain the command	ICC- command or CT-command
lenr	I/O-parameter, type IU16	Passing of the max. buffer size of the response field to the function and return of the actual length of the response in byte
response	Input parameter reference address of a field with elements of type IU8 in which the response is entered (field declaration in the application)	Response to the command

Tab. 5: Parameters for CT_data

When the function CT_data is called, the source address field usually contains the address value for 'Host'. In some applications, however, the address value for 'Remote Host' may occur. When a CT command is sent, the address value for the CardTerminal occurs as the destination address. When an ICC command is sent, the address value for the ICC has to be used. As a CardTerminal may have more than one ICC CardTerminal may have more than one ICC interface the address value for the respective ICC must be given. On the return of the response by the called function, the values in the parameters dad and sad are set respectively by the receiver and the sender.

Tab. 6 and 7 show the address values defined until now.

Sad in sedecimal notation (Hex)	Sender
02	HOST
05	REMOTE HOST
dad in sedecimal notation (Hex)	Receiver
00	ICC1 (IC card 1)
01	CT
02	ICC2 (IC card 2)
...	...
0E	ICC14 (IC card 14)
XX	other values reserved

Tab. 6: sad and dad values when sending commands

sad in sedecimal notation (Hex)	Sender
00	ICC1 (IC card 1)
01	CT
02	ICC2 (IC card 2)
...	...
0E	ICC14 (IC card 14)
dad in sedecimal notation (Hex)	Receiver
02	HOST
05	REMOTE HOST
XX	other values reserved

Tab. 7: sad and dad values when receiving responses

Note:
ICC1 has the value ‘00’ and not the value ‘01’ due to compatibility requirements with CT-API version 1.0.

Function values:

Parameter type	Meaning
Output parameter, Typ IS_8	Function values see section 4.4

Tab 8: Function values of CT_data

Notes for the user:

a) CardTerminal commands

The CT-data function allows the use of both inter-industry commands and manufacturer specific commands.

b) Commands to memory IC cards

If the CardTerminal supports communication with memory ICC, the commands shall comply with the designated conventions (e.g. use of inter-industry commands, if a corresponding transformation function to chip specific commands is available).

c) Commands to Processor IC Cards

Commands to processor ICC are passed through to the ICC in a transparent mode (i.e. without any changes). Implementation overheads (e.g. transmission protocol) are added automatically.

Fig. 1 in Annex D shows the transport of an ICC command and its response using the CT-API function CT_data.

4.3.3 CT_close

The function CT_close is the counterpart to the function CT_init. It terminates the communication with the CardTerminal which has been assigned to a logical CardTerminal number by the function CT_init.
The function shall be called before the end of the program in order to free resources if necessary.

Function:

CT_close (ctn)

Function parameter:

Parameter name	Parameter type	Meaning
ctn	Input-parameter, type IU 16	Logical CardTerminal number

Tab. 9: Parameters for CT_close

Function values:

Parameter type	Meaning
Output parameter, Typ IS8	Function values see section 4.4

Tab 10: Function values of CT_close

4.4 General function values of the CT-API functions

The CT-API functions return the function values listed in tab. 11. The function values are generally of the type IS8.
Return codes should be (as long as supported by the programming language) implemented in the form of global, typified constants ('OK', 'ERR_INVALID', etc.).

Return code	Function value	Meaning
OK	0	Function call was successful
ERR_INVALID	-1	Invalid parameter or value
ERR_CT	-8	CT error [1)]
ERR_TRANS	-10	Non-eliminable transmission error[2)]
ERR_MEMORY	-11	Memory assignment error in HTSI[3)]
ERR_HOST	-127	Abort of function by host/OS
ERR_HTSI	-128	HTSI error

Tab. 11: Function values of the CT- API functions

[1)] The CT is temporarily not accessible (busy with other or internal processes). The problem can be solved by the application.

[2)] Transmission errors due to mechanical, electrical or protocol failures. Reset of the CT is necessary.

[3)] A memory error occurred (e.g. the allocated buffer (memory) is too small for the data set).

Subsequent actions in error situations shall be implemented in accordance with HTSI manufacturers' recommendations.

Annex A (normative)

Identification of CT API Functions with a 'well known identifier'

In a development environment in which the functions of the various CT-API libraries shall be identifiable, a 'well known identifier' (wki) should be added to the function names.
The wki consists of
- the CardTerminal manufacturer Id (CTM Id, 5 bytes) and
- the HTSI Id (2 bytes)

Thus the CT-API functions have the form

CTxxxxxyy_init
CTxxxxxyy_data
CTxxxxxyy_close

whereby xxxxx depicts the CTM Id and yy the HTSI Id. The CTM ID is issued in co-operation with the CT manufacturer and registered by the RID German National Registration Authority. It consists of a 2 byte long country code in alpha coding according to ISO 3166 (e.g. DE for Germany) and a 3 byte long alphanumerical manufacturer code. The HTSI Id is assigned by the CT manufacturer.

Fig. 1 shows the general construction principle of an application system with two CT-API libraries and two CardTerminals.

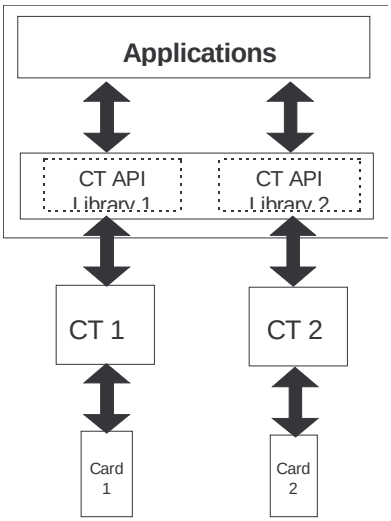

Fig.1: Application system with two different CT-API libraries and CardTerminals

Identification of CT API file names with a 'well known identifier'

If it is necessary to have a significant separation of software modules from different manufacturers (dynamic link libraries, device driver etc.), also a 'well known identifier' (wki) should be added to the file names.
In this case the wki consists of
- the CardTerminal manufacturer Id (CTM Id, 3 bytes) and
- the individual part (0-3 bytes)

Thus the CT-API file names have the form

ctxxx[yyy]

whereby xxx depicts the CTM Id and yyy the individual part. The CTM ID is issued in co-operation with the CT manufacturer and registered by the RID German National Registration Authority. It consists of a 3 byte long alphanumerical manufacturer code. The remaining part is optional and may be used individually by the CT manufacturer.

Annex B (informative)

Information for Programmers

Windows 16 Bit DLL

The functions has to be declared in pascal convention and with ”far” calls.

Also the parameters of pointers have to be declared as ”far”.

Annex C (informative)

Programming Example

The following example shows the use of CT_API functions in the programming language ANSI C

```
#include  <stdio.h>
#include  <conio.h>
#include  "ct_api.h"

#define   MAXMEM   1000

int main(void)
{
  unsigned char sad, dad;                  /* source / destination address, */
                                           /* declared as byte  */
  unsigned char command[300];              /* field for command with max. 300 characters length */
  unsigned char response [MAXMEM];         /* field for function response, here: max. 1000 bytes */
  unsigned short int ctn, lenr;            /* Card Terminal number, response length */
  char res                                 /*function value */

  ctn=1;                                   /* Card Terminal 1 */

  /* select logical Card Terminal number 1 and port COM2  (dependent on manufacturer!)*/
  if (CT_init(ctn,2) != OK) return(1);     /* if return code not OK, end program */

  /* CT command REQUEST ICC (20 12 01 00 00 ) construct and transmit */
  prinff ("\nPlease insert card and press any key!\n");   getch ( );

  sad=2;                                   /* source = host */
  dad=1;                                   /* destination = CardTerminal */
  lenr=MAXMEM;                             /* maximum response length as Info to the function */
  command[0]=0x20;                         /* CLA */
  command[1]=0x12;                         /* INS */
  command[2]=0x01;                         /* P1 */
  command[3]=0x00;                         /* P2 */
  command[4]=0x00;                         /* L */

  /* Call function (CT_data) */

..res=CT_data (ctn,&dad,&sad,5,command,&lenr,response);

  /* Error handling not described here */
                    ...
  /* after card ejection with CT command EJECT ICC */
  res=CT_close(ctn);

  return(0);
}
```

Annex D (informative)

Example of an internal processing of the CT-API function CT_data

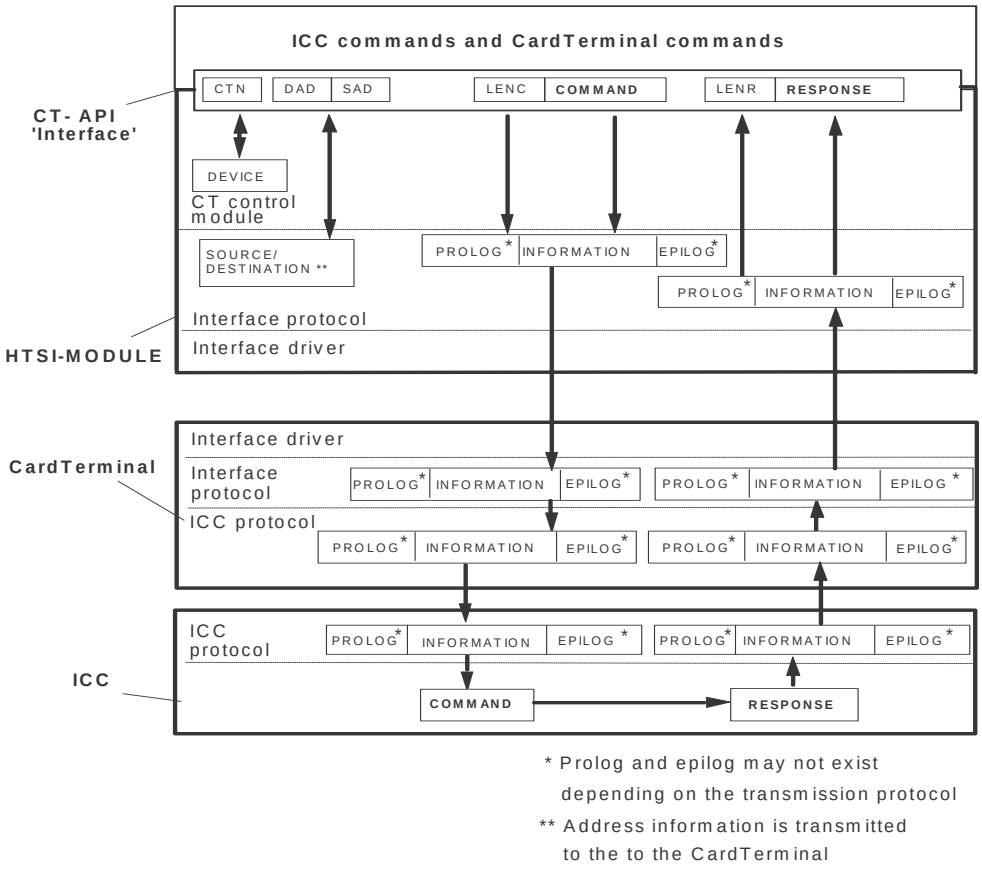

Fig.1: Example of the internal processing of the CT-API Function CT_data