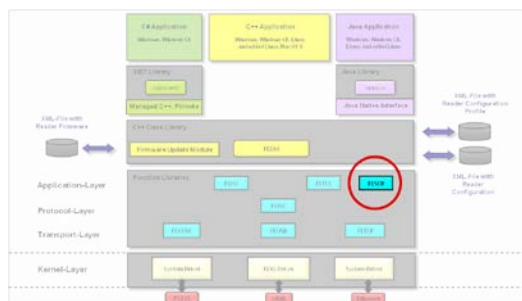


ID FESCR

Version 1.02.01

ISO7816 smartcard Interface for OBID® *classic-pro* Reader



Operating System	Target		Notes
	32-Bit	64-Bit	
Windows Vista / 7 / 8 / 10	X	X	
Windows CE 6	X	-	On request
Linux	X	X	
Android	X	X	On request
Apple Max OS X	-	X	On request

Note

© Copyright 2011-2014 by FEIG ELECTRONIC GmbH
Lange Straße 4
D-35781 Weilburg-Waldhausen
Germany
Tel.: +49 6471 3109-0
<http://www.feig.de>

The indications made in these mounting instructions may be altered without previous notice. With the edition of these instructions, all previous editions become void.

Copying of this document, and giving it to others and the use or communication of the contents thereof, are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a patent or the registration of a utility model or design.

Composition of the information given in these mounting instructions has been done to the best of our knowledge. FEIG ELECTRONIC GmbH does not guarantee the correctness and completeness of the details given and may not be held liable for damages ensuing from incorrect installation.

Since, despite all our efforts, errors may not be completely avoided, we are always grateful for your useful tips.

FEIG ELECTRONIC GmbH assumes no responsibility for the use of any information contained in this manual and makes no representation that they are free of patent infringement. FEIG ELECTRONIC GmbH does not convey any license under its patent rights nor the rights of others.

The installation-information recommended here relates to ideal outside conditions. FEIG ELECTRONIC GmbH does not guarantee the failure-free function of the OBID®-system in outside environment.

OBID® and OBID *i-scan*® are registered Trademarks of FEIG ELECTRONIC GmbH.

Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

Windows Vista is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries

Linux® is a registered Trademark of Linus Torvalds.

Licensing agreement for use of the software

This is an agreement between you and FEIG ELECTRONIC GmbH (hereafter "FEIG") for use of the ID FESCR program library and the present documentation, hereafter called licensing material. By installing and using the software you agree to all terms and conditions of this agreement without exception and without limitation. If you are not or not completely in agreement with the terms and conditions, you may not install the licensing material or use it in any way. This licensing material remains the property of FEIG ELECTRONIC GmbH and is protected by international copyright.

§1 Object and scope of the agreement

1. FEIG grants you the right to install the licensing material provided and to use it under the following conditions.
2. You may install all components of the licensing material on a hard disk or other storage medium. The installation and use may also be done on a network fileserver. You may create backup copies of the licensing material.
3. FEIG grants you the right to use the documented program library for developing your own application programs or program libraries, and you may sell the runtime file FESCR.DLL or LIBFESCR.SO.x.y.z¹ without licensing fees under the stipulation that these application programs or program libraries are used to control devices and/or systems which are developed and/or sold by FEIG.

§2. Protection of the licensing material

1. The licensing material is the intellectual property of FEIG and its suppliers. It is protected in accordance with copyright, international agreements and relevant national statutes where it is used. The structure, organization and code of the software are a valuable business secret and confidential information of FEIG and its suppliers.
2. You agree not to change, modify, translate, reverse develop, decompile, disassemble the program library or the documentation or in any way attempt to discover the source code of this software.
3. To the extent that FEIG has applied protection marks, such as copyright marks and other legal restrictions in the licensing material, you agree to keep these unchanged and to use them unchanged in all complete or partial copies which you make.
4. The transmission of licensing material in part or in full is prohibited unless there is an explicit agreement to the contrary between you and FEIG. Application programs or program libraries which are created and sold in accordance with §1 Par. 3 of this Agreement are excepted.

§3 Warranty and liability limitations

1. You agree with FEIG that it is not possible to develop EDP programs such that they are error-free for all application conditions. FEIG explicitly makes you aware that the installation of a new program can affect already existing software, including such software that does not run at the same time as the new software. FEIG assumes no liability for direct or indirect damages, for consequential damages or special damages, including lost profits or lost savings. If you want to ensure that no already installed program will be affected, you should not install the present software.
2. FEIG explicitly notes that this software makes it possible for irreversible settings and adaptations to be made on devices which could destroy these devices or render them unusable. FEIG assumes no liability for such actions, regardless of whether they are carried out intentionally or unintentionally.
3. FEIG provides the software „as is“ and without any warranty. FEIG cannot guarantee the performance or the results you obtain from using the software. FEIG assumes no liability or guarantee that the protection rights of third parties are not violated, nor that the software is suitable for a particular purpose.
4. FEIG call explicit attention the licensed material is not designed with components and testing for a level of reliability suitable for use in or in connection with surgical implants or as critical components in any life support systems whose failure to perform can reasonably be expected to cause significant injury to a human.
To avoid damage, injury, or death, the user or application designer must take reasonably prudent steps to protect against system failures.

¹ x.y.z represents the actual version number

§4 Concluding provisions

1. This Agreement contains the complete licensing terms and conditions and supercedes any prior agreements and terms. Changes and additions must be made in writing.
2. If any provision this agreement is declared to be void, or if for any reason is declared to be invalid or of no effect, the remaining provisions shall be in no manner affected thereby but shall remain in full force and effect. Both parties agree to replace the invalid provision with one which comes closest to its original intention.
3. This agreement is subject to the laws of the Federal Republic of Germany. Place of jurisdiction is Frankfurt a. M.

Contents:

1. Introduction.....	6
1.1. Shipment.....	7
1.1.1. Windows Vista / 7 / 8 / 10	8
1.1.2. Windows CE 6	9
1.1.3. Linux	10
2. Changes since the previous version	11
3. Installation.....	12
3.1. 32- and 64-Bit Windows Vista / 7 / 8 / 10.....	12
3.2. Windows CE 6	13
3.3. 32- and 64-Bit Linux	14
4. Incorporating into the application program	15
4.1. Supported Development Tools.....	15
4.2. Incorporating into Visual Studio	15
5. Programming Interface	16
5.1. Overview	16
5.2. List of functions	18
5.2.1. FESCR_NewSmartCardSlot	19
5.2.2. FESCR_DeleteSmartCardSlot	20
5.2.3. FESCR_GetSmartCardSlotList	21
5.2.4. FESCR_GetDLLVersion	22
5.2.5. FESCR_GetErrorText	23
5.2.6. FESCR_GetLastError	24
5.2.7. FESCR_Activate	25
5.2.8. FESCR_Deactivate	26
5.2.9. FESCR_APDU	27
6. Appendix	28
6.1. Error codes	28
6.2. History.....	29

1. Introduction

The support package ID FESCR is intended to support in programming ISO 7816 (contact) based application software, integrate the OBID® *classic-pro* Reader, and supports ANSI-C, ANSI-C++ and essentially any other language which can invoke C functions.

The support package provides a simple ISO 7816 function interface for easy APDU exchange and is designed to work together with the FEISC.DLL for the OBID i-scan® and OBID® *classic-pro* Reader.

The support package ID FECOM can be used for communication through a serial port on the PC. Communication through a USB port requires the support package ID FEUSB and communication through an Ethernet port (TCP/IP) requires the support package ID FETCP.

This library package can be used with the following Operating Systems:

Operating System	Target		Notes
	32-Bit	64-Bit	
Windows Vista / 7 / 8 / 10	X	X	
Windows CE 6	X	-	On request
Linux	X	X	
Android	X	X	On request
Apple Max OS X	-	X	On request

1.1. Shipment

This support package consists of files listed in the tables below. Normally, this package is shipped together with other libraries in a Software Development Kit (SDK) – e.g. ID ISC.SDK.Win.

1.1.1. Windows Vista / 7 / 8 / 10

File	Use
FESCR.DLL	DLL with all functions
FESCR.LIB	LIB file for linking with C/C++ projects
FESCR.H	Header file for C/C++ projects

1.1.2. Windows CE 6

File	Use
FESCRCE.DLL	DLL with all functions
FESCRCE.LIB	LIB file for linking with C/C++ projects
FESCR.H	Header file for C/C++ projects

1.1.3. Linux

File	Use
LIBFESCR.SO.x.y.z ²	Function library
FESCR.H	Header file for C/C++ projects

² x.y.z. represents the version number of the library file

2. Changes since the previous version

- bugfix for BlockID calculation
- Linux:
1st Version for 64-Bit
- Discontinued support for Windows XP, Windows CE 4 and CE 5

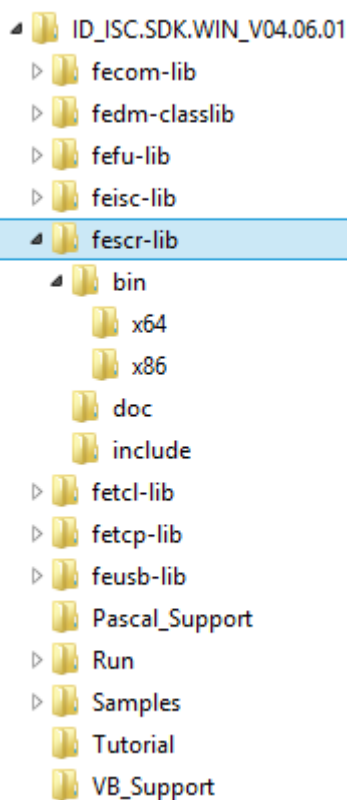
Please note also the revision history in the Appendix to this document.

3. Installation

Normally, this package is shipped together with other libraries in a Software Development Kit (SDK). Copy the SDK into a directory of your choice.

The files of this library package can be found in the sub-directory fescr-lib.

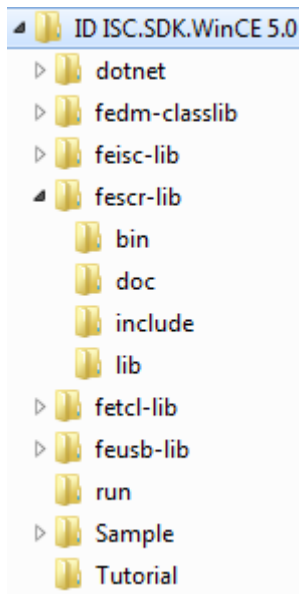
3.1. 32- and 64-Bit Windows Vista / 7 / 8 / 10



If you won't add your projects to the Samples path, we recommend the following steps:

- Copy FESCR.DLL into the directory of the application program (recommended) or into the Windows system directory.
- Copy FESCR.LIB into the project or LIB directory.
- Copy FESCR.H into the project or INCLUDE directory.

3.2. Windows CE 6

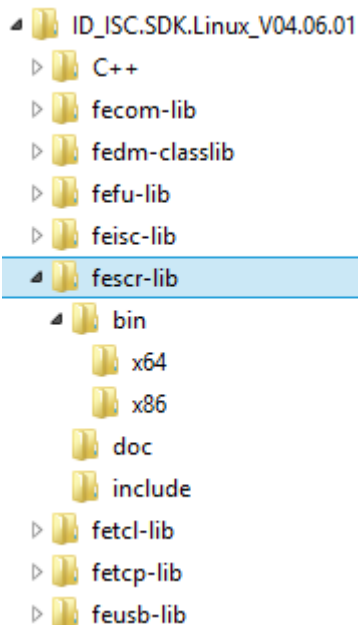


If you won't add your projects to the Samples path, we recommend the following steps:

- Copy FESCRCE.DLL into the system directory of the Windows CE system.
- Copy FESCRCE.LIB into the project or LIB directory.
- Copy FESCR.H into the project or INCLUDE directory

Note: you cannot use the DLL together with eMbedded Visual Basic 3.0.

3.3. 32- and 64-Bit Linux



Choose one option for installation:

Option 1: If an install.sh is shipped inside the SDK root directory, execute this install script. It will copy all library files into the directory /usr/lib resp. /usr/lib64 and creates symbolic links for each library file. The header file can be copied into a directory of your choice.

Option 2: Copy all files of this support package into a directory of your choice and create symbolic links for libfecscr.so.x.y.z³ in the directory /usr/lib resp. /usr/lib64 with the following calls:

cd /usr/lib (for 64 Bit : /usr/lib64)

In -s /<your_directory>/libfecscr.so.x.y.z libfecscr.so.x

In -s /<your_directory>/libfecscr.so.x libfecscr.so

ldconfig

The compiled library is linked against LibC V6 und LibStdC++ V6.

³ x.y.z represents the version number

4. Incorporating into the application program

4.1. Supported Development Tools

Operating System	Development Tool	Supported
Windows Vista / 7 / 8 / 10	Visual Studio	Yes
	Borland C++ Builder	Yes
	Embarcadero C++ Builder	Yes
Windows CE 6	eMbedded Visual C++ 4	No
	Visual Studio 2005	No
	Visual Studio 2008	Yes
Linux	GCC	Yes
Mac OS X	GCC	Yes, for projects with x86_64 architecture
	Xcode	Yes, for projects with x86_64 architecture

For C/C++ programmers:

If the LIB file (only Windows) was made known to the development tool, any function may be immediately used. This presumes of course the declaration of the DLL functions with an #include instruction within each source file that invokes FESCR functions.

ID FEISC and one of the packages ID FECOM and/or ID FEUSB and/or ID FETCP must also be incorporated into your project.

4.2. Incorporating into Visual Studio

1. Add Include path for the header file in project settings (category C/C++)
2. Add fescl.lib (optional with path) in project settings (category Linker)

5. Programming Interface

5.1. Overview

The FESCR library encapsulates for the programmer all the functions and parameters necessary for simple ISO 7816 (contact) based communication with ISO7816 compliant smartcards, accessed by a reader of the OBID® *classic-pro* Reader Family. Together with the support packages ID FEISC and one of ID FECOM, ID FETCP or ID FEUSB, this makes it possible to handle complex ISO 7816 commands.

The functions in FESCR are responsible for internal administration, activation and deactivation of smartcards, T=0 and T=1 protocol cycle handling, T=0 and T=1 response data collection and any necessary error outputs.

The FESCR library alone is not enough to communicate with an OBID® *classic-pro* Reader. You can however initiate the communication process and use the FEISC and one of the port libraries (FECOM, FEUSB, FETCP) to communicate with an OBID® *classic-pro* Reader over an asynchronous serial interface or with a TCP/IP-Server or through the USB port. Other interface drivers can be integrated with the Plug-In mechanism of the FEISC.

Use of the FEUSB for communicating with OBID® USB devices is mandatory.

The core elements of the library are the Object Manager and the smartcard slot objects generated during runtime. Before activation it is mandatory to create a smartcard slot object for every smartcard slot. Every smartcard slot object represents an ISO7816 compliant smartcard. After remove of the smartcard, the smartcard slot object must be deleted manually by the application.

The SCR Object Manager implements self-administration which frees an application program from having to buffer any values, parameters or other settings: It keeps a list with all generated smartcard slot objects. The smartcard slot object is the central program section that carries out the communication cycles. Each smartcard slot object administers all the parameters relevant to its protocol tasks within its local memory and are independent of each other. However, for every OBID® *classic-pro* Reader the application can start only one APDU communication cycle at the same time.

Before first activation you must create a smartcard slot object using the **FESCR_NewSmartCardSlot** function. If this is done without error, the return value includes a handle which is used by the application program as an access number. This handle is required for unique identification of the generated smartcard slot object. If you are using self-administration, the smartcard slot Object List can be called up using the **FESCR_GetSmartCardSlotList** function.

A smartcard slot object generated using **FESCR_NewSmartCardSlot** must always be deleted from memory using the **FESCR_DeleteSmartCardSlot** function.

If an application program is opened multiple times, each program (instance) gets an empty object list by invoking **FESCR_GetSmartCardSlotList**. This prevents mixing up access rights under different program instances.

Nearly all the library functions have a return value which is negative in case of error.

5.2. List of functions

The support package contains functions for various tasks. They are divided into groups for better orientation.

Administration functions for SmartCard Slot-Objects

- **int FESCR_NewSmartCardSlot**(int iReaderHnd, unsigned char ucBusAdr, int iSlotNo, unsigned char ucNad, bool bUseNad)
- **int FESCR_DeleteSmartCardSlot**(int iSlotHnd)
- **int FESCR_GetSmartCardSlotList**(int iNext)

Query functions

- **void FESCR_GetDLLVersion**(char* cVersion)
- **int FESCR_GetErrorText**(int iErrorCode, char* cErrorText)
- **int FESCR_GetLastError**(int iSlotHnd, int* iErrorCode, char* cErrorText)

Special communication functions

- **int FESCR_Activate**(int iSlotHnd, unsigned char* ucReqData, int iReqDataLen, unsigned char* ucRspData, unsigned int uiRspDataBufSize, int* iRspDataLen)
- **int FESCR_Deactivate**(int iSlotHnd)
- **int FESCR_Apdu**(int iSlotHnd, unsigned char* ucReqApdu, int iReqApduLen, unsigned char* ucRspApdu, unsigned int uiRspApduBufSize, int* iRspApduLen)

5.2.1. FESCR_NewSmartCardSlot

Function	Creates a smartcard slot Object.
Syntax	int FESCR_NewSmartCardSlot(int iReaderHnd, unsigned char ucBusAdr, int iSlotNo, unsigned char ucNad, bool bUseNad)
Description	<p>A smartcard slot object is created. Communications based on APDUs require a smartcard slot Object in order to run.</p> <p>Multiple smartcard slot Objects can in principle carry out their communication over the same Reader Object, if the communication protocols are scheduled successive.</p> <p>A smartcard slot Object created with FESCR_NewSmartCardSlot must (!) be deleted from memory using the FESCR_DeleteSmartCardSlot function. Otherwise the memory reserved by the library is not freed up again.</p> <p><i>iReaderHnd</i> is the handle of a Reader Object created from FEISC using the FEISC_NewReader function.</p> <p><i>ucBusAdr</i> is the bus address of the Reader. For USB Reader, this parameter has no impact.</p> <p><i>iSlotno</i> is the physical smartcard slot, which should be used for communication</p> <p><i>ucNad</i> is the card identifier and is used internally, if <i>bUseNad</i> is true.</p>
Return value	<p>If a smartcard slot Object was created without error, a handle (>0) is returned. In case of error, the function returns a value less than zero.</p> <p>A list of error codes can be found in the Appendix.</p>
Example	<pre> ... #include "fecom.h" #include "feisc.h" #include "fescr.h" char cPortNr[4]; itoa(1, cPortNr, 10); // Convert Integer to Char int iPortHnd = FECOM_OpenPort(cPortNr); // COM:1 should be opened if(iPortHnd < 0) { // code here in case of error } else { // Open Reader object int iReaderHnd = FEISC_NewReader(iPortHnd); if(iReaderHnd > 0) { int iSlotHnd = FESCR_NewSmartCardSlot(iReaderHnd, 255, 1, 0, false); } } </pre>

5.2.2. FESCR_DeleteSmartCardSlot

Function	Deletes a smartcard slot object
Syntax	int FESCR_DeleteSmartCardSlot(int iSlotHnd)
Description	The function deletes the smartcard slot Object indicated by the parameter <i>iSlotHnd</i> and frees up the reserved memory.
Return value	The return value is 0 if the action was successful. In case of error, the function returns a value less than zero. A list of error codes can be found in the Appendix.
Example	<pre>... #include "feisc.h" #include "fescr.h" int iErr; int iReaderHnd = FEISC_NewReader(iPortHnd); if(iReaderHnd < 0) { // code here in case of error } if(iReaderHnd > 0) { int iSlotHnd = FESCR_NewSmartCardSlot(iReaderHnd, 255, 1, 0, false); ... iErr = FESCR_DeleteSmartCardSlot(iSlotHnd); }</pre>

5.2.3. FESCR_GetSmartCardSlotList

Function	Depending on the <i>iNext</i> parameter, gets the first or following smartcard slot handle from the internal list of the generated smartcard slot Objects.
Syntax	int FESCR_GetSmartCardSlotList(int iNext)
Description	The function returns a smartcard slot handle from the internal list of smartcard slot handles. If one transmits a 0 for <i>iNext</i> , the first entry in the list is returned. If you transmit a smartcard slot handle contained in the list with <i>iNext</i> , the function gets and returns the entry following the smartcard slot handle. In this way you can keep incrementing the return value to go through the list and call out all the entries.
Return value	<p>When an entry is found, the smartcard slot handle is provided with the return value. When the end of the internal list is reached, in other words the transferred smartcard slot handle has no following entry, a 0 is returned. If there is no smartcard slot Object, FESCR_ERR_EMPTY_LIST is returned.</p> <p>In case of error, the function returns a value less than zero.</p> <p>A list of error codes can be found in the Appendix.</p>
Example	<pre> ... #include "fescr.h" ... // Example function for creating a list of smartcard slot objects void SmartCardSlotList(void) { int iNextHnd = FESCR_GetSmartCardSlotList(0); // get the first handle while(iNextHnd > 0) { // here for example code for collecting the handles and reading out parameters ... iNextHnd = FESCR_GetSmartCardSlotList(iNextHnd); // get next handle } ... // here for example code for displaying a list } </pre>
Tip	<p>When closing all open created smartcard slot Objects it is convenient to use a loop such as in the example above. Bear in mind however than you cannot get the next in line from a deleted smartcard slot Object. The following code fragment gives you an idea of how to delete all created smartcard slot Objects in a loop:</p> <pre> ... int iNextHnd, iCloseHnd, iError; iNextHnd = FESCR_GetSmartCardSlotList(0); // get first handle while(iNextHnd > 0) { iCloseHnd = iNextHnd; iNextHnd = FESCR_GetSmartCardSlotList(iNextHnd); // get next handle iError = FESCR_DeleteSmartCardSlot(iCloseHnd);// only now delete smartcard slot Object } </pre>

5.2.4. FESCR_GetDLLVersion

Function	Gets the DLL/SO version number.
Syntax	void FESCR_GetDLLVersion(char* cVersion)
Description	<p>The function returns the version number of the DLL/SO.</p> <p><i>cVersion</i> is an empty, null-terminated string for returning the version number. The string should be able to hold at least 256 characters.</p> <p>The string is filled with the current version number (e.g. „01.02.01“). Newer versions may provide additional information.</p>
Return value	none
Example	<pre>... #include "fescr.h" char cVersion[256]; FESCR_GetDLLVersion(cVersion); // code here for displaying the version number</pre>

5.2.5. FESCR_GetErrorText

Function	Gets error text for error code
Syntax	int FESCR_GetErrorText(int iErrorCode, char* cErrorText)
Description	This function uses <i>cErrorText</i> to send a short error text associated with the <i>iErrorCode</i> . The buffer for <i>cErrorText</i> should be able to hold at least 256 characters.
Return value	If there is no error the function returns zero, and if error a value less than zero. The list of error codes can be found in the Appendix.
Example	<pre>... #include "fescr.h" char cErrorText[256]; ... int iBack = FESCR_GetErrorText(FESCR_ERR_EMPTY_LIST, cErrorText) // code here for displaying the text</pre>

5.2.6. FESCR_GetLastError

Function	Gets the last error code and transmits error text.
Syntax	int FESCR_GetLastError(int iSlotHnd , int* iErrorCode, char* cErrorText)
Description	<p>The function uses <i>iErrorCode</i> to send the last error code of the smartcard slot object selected with <i>iSlotHnd</i> and transmits the associated error text in <i>cErrorText</i>.</p> <p>The buffer for <i>cErrorText</i> should be able to hold at least 256 characters.</p>
Return value	If no error the function returns zero, and in case of error a value less than zero. A list of error codes can be found in the Appendix.
Example	<pre>... #include "fescr.h" char cErrorText[256]; int iErrorCode = 0; ... int iBack = FESCR_GetLastError(iSlotHnd, &iErrorCode, cErrorText) // code here for displaying the text</pre>

5.2.7. FESCR_Activate

Function	Function activates a smartcard.
Syntax	<pre>int FESCR_Activate(int iSlotHnd, unsigned char* ucReqData, int iReqDataLen, unsigned char* ucRspData, unsigned int uiRspDataBufSize, int* iRspDataLen)</pre>
Description	<p>This function activates a smartcard.</p> <p><i>iSlotHnd</i> is the handle for the smartcard slot Object.</p> <p><i>ucReqData</i> is a pointer to a buffer with the request data.</p> <p>(refer to the system manuals of the readers: [0xC0] [0x01] SAM Activate/Deactivate: <i>ucReqData must start with byte MODE</i>)</p> <p><i>iReqDataLen</i> contains the number of bytes in <i>ucReqData</i>.</p> <p><i>ucRspData</i> is a pointer to a buffer used for the response data.</p> <p><i>ucRspDataBufSize</i> contains the size of the <i>ucRspData</i> buffer</p> <p><i>iRspDataLen</i> contains the length of the response data.</p>
Return value	If no error the function returns zero, and in case of error a value less than zero. A list of error codes can be found in the Appendix.
Example	<pre>... #include "fescr.h" // Create New SmartCardSlot ... // Activate SmartCard unsigned char ucReqData[1] = {0x03}; // Activate Smartcard with T=1 protocol unsigned char ucRspData[256]; int iRspDataLen = 0; int iBack = FESCR_Activate(iSlotHnd, ucReqData, 1, ucRspData, sizeof(ucRspData), &iRspDataLen); ...</pre>

5.2.8. FESCR_Deactivate

Function	Function deactivates a smartcard.
Syntax	int FESCR_Deactivate(int iSlotHnd)
Description	This function deactivates a smartcard. <i>iSlotHnd</i> is the handle for the smartcard slot Object.
Return value	If no error the function returns zero, and in case of error a value less than zero. A list of error codes can be found in the Appendix.
Example	<pre>... #include "fescr.h" // Create New SmartCardSlot ... // Activate SmartCard unsigned char ucReqData[1] = {0x03}; // Activate Smartcard with T=1 protocol unsigned char ucRspData[256]; int iRspDataLen = 0; int iBack = FESCR_Activate(iSlotHnd, ucReqData, 1, ucRspData, sizeof(ucRspData), &iRspDataLen); // Deactivate SmartCard Slot int iBack = FESCR_Deactivate(iSlotHnd); ... </pre>

5.2.9. FESCR_APDU

Function	Function executes an APDU.
Syntax	<pre>int FESCR_APDU(int iSlotHnd, unsigned char* ucReqApdu, int iReqApduLen, unsigned char* ucRspApdu, unsigned int uiRspApduBufSize, int* iRspApduLen)</pre>
Note	<p>This function handles the APDU communication process.</p> <p>One smartcard slot Object can handle only one APDU command at the same time. Multiple APDU communication processes can be started, if each APDU is handled by a different Reader, which implies the use of multiple Reader Objects in the FEISC library.</p> <p><i>iSlotHnd</i> is the handle for the smartcard slot Object.</p> <p><i>ucReqApdu</i> is a pointer to a buffer with the ADPU command.</p> <p><i>iReqApduLen</i> contains the number of bytes in <i>ucReqApdu</i>.</p> <p><i>ucRspApdu</i> is a pointer to a buffer used for the response APDU.</p> <p><i>uiRspApduBufSize</i> contains the size of the <i>ucRspApdu</i> buffer.</p> <p><i>iRspApduLen</i> contains the length of the response APDU.</p>
Return value	If no error the function returns zero, and in case of error a value less than zero. A list of error codes can be found in the Appendix.
Example	<pre>... #include "fescr.h" ... int MyApu() { //APDU response buffer unsigned char ucRspApduData[65536]; int iRspApduLen = 0; unsigned char* pucApu ; ... // build the APDU, allocate memory and save it in pucApu. ... // execute the APDU. The function returns immediately return FESCR_APDU(iSlotHnd, pucApu, iApuLen, ucRspApduData, sizeof(ucRspApduData), & iRspApduLen); }</pre>

6. Appendix

6.1. Error codes

Error constants	Value	Description
FESCR_ERR_NEW_SMARTCARDSLOT_FAILURE	-4300	Error in creating a new smartcard slot Object
FESCR_ERR_EMPTY_LIST	-4301	Smartcard slot handle list is empty (no smartcard slot Objects stored)
FESCR_ERR_POINTER_IS_NULL	-4302	Pointer to transfer parameter is NULL
FESCR_ERR_NO_MORE_MEM	-4303	No more system memory
FESCR_ERR_NO_USB_SUPPORT	-4006	No USB support (e.g. under NT4)
FESCR_ERR_APDU_CURRENTLY_RUNNING	-4309	An APDU is still running and not finished yet
FESCR_ERR_UNKNOWN_HND	-4320	The transferred smartcard slot handle is unknown
FESCR_ERR_HND_IS_NULL	-4321	The transferred smartcard slot handle is 0
FESCR_ERR_HND_IS_NEGATIVE	-4322	The transferred smartcard slot handle is negative
FESCR_ERR_NO_HND_FOUND	-4323	No smartcard slot handle found in smartcard slot handle list
FESCR_ERR_READER_HND_IS_NEGATIVE	-4326	The transferred Reader handle is negative
FESCR_ERR_THREAD_NOT_CREATED	-4327	The APDU thread could not be started
FESCR_ERR_UNKNOWN_PARAMETER	-4350	Transfer parameter is unknown
FESCR_ERR_PARAMETER_OUT_OF_RANGE	-4351	Transfer parameter too large or too small
FESCR_ERR_UNKNOWN_ERRORCODE	-4353	Unknown error code
FESCR_ERR_UNDERSIZED_RESPONSE_BUFFER	-4357	The response buffer is too small
FESCR_INVALID_ACKNOWLEDGEMENT_LENGTH	-4373	Too less response data from reader after APDU transmission
FESCR_LIST_COMPLETE_FAILURE	-4374	Internal error in front of APDU transmission
FESCR_INCOMPLETE_RESPONSE	-4375	Receive procedure interrupt causes too less response data
FESCR_INVALID_PROTOCOL	-4376	Unknown command or invalid protocol data
FESCR_INVALID_TRANSMISSION	-4377	Internal error in FEISC library in front of transmission or invalid reader status after transmission.

6.2. History

V1.00.00

- Initial release