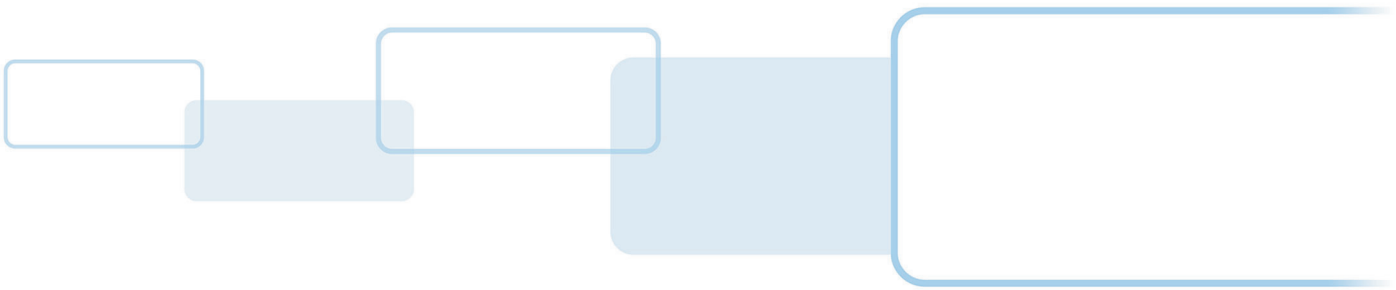
SYNCHRONOUS-API
SOFTWARE DEVELOPER GUIDE

PLT-03635, Rev. A.0

January 2018



hidglobal.com

Copyright

Trademarks

Revision History

Date	Description	Revision
January 2018	Initial Release.	A.0

Contacts

For additional offices around the world, see www.hidglobal.com/contact/corporate-offices

Americas and Corporate

611 Center Ridge Drive
Austin, TX 78753
USA
Phone: 866 607 7339
Fax: 949 732 2120

Asia Pacific

19/F 625 King's Road
North Point, Island East
Hong Kong
Phone: 852 3160 9833
Fax: 852 3160 4809

Europe, Middle East and Africa (EMEA)

Haverhill Business Park Phoenix Road
Haverhill, Suffolk CB9 7AE
England
Phone: 44 (0) 1440 711 822
Fax: 44 (0) 1440 714 840

Brazil

Condomínio Business Center
Av. Ermano Marchetti, 1435
Galpão A2 - CEP 05038-001
Lapa - São Paulo / SP
Brazil
Phone: +55 11 5514-7100

HID Global Technical Support: www.hidglobal.com/support

Contents

This page is intentionally left blank.

Section 1

Introduction

1.1 Release notes for version 3.0.0

Synchronous-API version 3.0.0 is a full release for x86 32-bit and x64 64-bit Windows operating systems. The following changes have been made:

- Added support for new AVR controller chip based contact readers.
- New .chm help file instead of .hlp file.
- New 32-bit and 64-bit installers.

1.2 Supported operating systems

Synchronous-API v3.0.0 supports the following operating systems:

- Windows XP SP3. Support is limited to 8051 controller based readers only. AVR controller based readers do not support Windows XP systems.
- Windows 7
- Windows 8
- Windows 8.1
- Windows 10
- Windows Server 2016

1.3 Important notes for developers

Because the functions of this API are similar to the functions of the Microsoft resource manager, you should carefully read the MSDN help for the following functions:

- `SCardEstablishContext`
- `SCardReleaseContext`
- `SCardListReaders`
- `SCardConnect`
- `SCardDisconnect`
- `SCardReconnect`

Because PC/SC supports only T=0 and T=1 smart cards, synchronous smart cards are handled as T=0 smart cards for the `SCardConnect` function, to fit into the given architecture. Therefore the `SCardConnect` function will succeed only if T=0 is used as the preferred protocol. Furthermore, the returned ATR is a valid T=0 ATR (simulated by the API). If you want to get the real ATR of the synchronous card, use only the last 4 bytes.

Note: I2C cards do not return an ATR during power on. To fit into the PC/SC architecture, a pseudo-ATR (3B 04 49 32 43 2E) is always returned by this API if an I2C card is inserted.

1.4 Why does the HID OMNIKEY® Synchronous-API exist?

PC/SC 1.0 does not support any synchronous smart cards, although they are used for many applications. Therefore HID has developed the Synchronous-API to give application developers the ability to access synchronous smart cards. Furthermore, all necessary header files and libraries are included for application development. To speed up the development process, and to help get acquainted with the API, sample programs (binaries and source code) are provided.

This new version of Synchronous-API only exists for backwards compatibility reasons to support existing customer software. For new implementations based on HID Global OMNIKEY readers with an AVR based controller chip, please use the Pseudo-APDUs concept developed for these new readers, which allow an implementation with direct PC/SC commands. For details please refer to the *OMNIKEY Contact Smart Card Readers Software Developer Guide* (PLT-03099) available from the HID website.

1.5 Cards supported by the OMNIKEY reader

The Synchronous-API consists of three DLLs which provide functions for:

- 2 wire bus protocol cards (like SLE4432 and SLE4442).
- 3 wire bus protocol cards (like SLE4418 and SLE4428).
- I2C bus protocol cards.
- Intelligent 416 Bit EEPROM with internal PIN check cards: SLE4404.
- DataFlash cards: AT45DB041B.
- Contactless cards: HID iCLASS®.
- Secure memory cards: AT88SC153, AT88SC1608, AT88SC101, AT88SC102.

The Synchronous-API supports the following families of smart card readers:

- 8051 controller based platform.
- AVR controller based platform.

Note: The synchronous smart cards that are supported depend on your reader. Please refer to the respective OMNIKEY reader developer guide for further information. The PC/SC reader name can be obtained using the `SCardListReaders` function. For further details, refer to MSDN.

Note: Check that you have installed the latest PC/SC driver for your reader.

1.5.1 Supported readers: 8051 controller based platform

PC/SC reader name	Supported credentials
CardMan 3021	2WBP, 3WBP, I2C, 4404, AT45DB041B, AT88SC153, AT88SC1608, AT88SC101, AT88SC102
CardMan 3121	2WBP, 3WBP, I2C, 4404, AT45DB041B, AT88SC153, AT88SC1608, AT88SC101, AT88SC102
CardMan 3111	2WBP, 3WBP, I2C, 4404, AT45DB041B, AT88SC153, AT88SC1608, AT88SC101, AT88SC102
CardMan 3621	2WBP, 3WBP, I2C, 4404, AT45DB041B, AT88SC153, AT88SC1608, AT88SC101, AT88SC102
CardMan 3821	2WBP, 3WBP, I2C, 4404, AT45DB041B, AT88SC153, AT88SC1608, AT88SC101, AT88SC102
CardMan 6121	2WBP, 3WBP, I2C
CardMan 4040	2WBP, 3WBP, I2C, 4404, AT88SC153, AT88SC1608, AT88SC101, AT88SC102
CardMan 4321	2WBP, 3WBP, I2C, 4404, AT88SC153, AT88SC1608, AT88SC101, AT88SC102
CardMan 5x21	2WBP, 3WBP, I2C, 4404, AT88SC153, AT88SC1608, AT88SC101, AT88SC102, HID iCLASS
CardMan 6321	2WBP, 3WBP, I2C, 4404, AT88SC153, AT88SC1608, AT88SC101, AT88SC102
CardMan Smart@Link	2WBP, 3WBP, I2C, 4404, AT45DB041B, AT88SC153, AT88SC1608, AT88SC101, AT88SC102
CardMan Smart@Key	2WBP, 3WBP, I2C, 4404, AT88SC153, AT88SC1608, AT88SC101, AT88SC102
Serial Smart Card Reader	2WBP, 3WBP, I2C, 4404, AT45DB041B, AT88SC153, AT88SC1608, AT88SC101, AT88SC102
PC-Card Smart Card Reader	2WBP, 3WBP, I2C, 4404, AT88SC153, AT88SC1608, AT88SC101, AT88SC102
USB CCID Smart Card Reader	2WBP, 3WBP, I2C, 4404, AT45DB041B, AT88SC153, AT88SC1608, AT88SC101, AT88SC102

1.5.2 Supported readers: AVR controller based platform

PC/SC reader name	Supported credentials
OMNIKEY 3021	2WBP, 3WBP, I2C
OMNIKEY 3121	2WBP, 3WBP, I2C
OMNIKEY 6121	2WBP, 3WBP, I2C
OMNIKEY 5x22	2WBP, 3WBP, I2C
OMNIKEY Smart@Link	2WBP, 3WBP, I2C
Cherry KBD KC 1000 SC	2WBP, 3WBP, I2C
Fujitsu KB100 SCR	2WBP, 3WBP, I2C
CHERRY SMART TERMINAL XX44	2WBP, 3WBP, I2C
Generic USB Smart Card Reader	2WBP, 3WBP, I2C
Fujitsu Smart Card Reader D323	2WBP, 3WBP, I2C

1.6 Getting the ATR (Answer to Reset)

Note: Before you can get the ATR or other attributes you must connect to the card.

You get the ATR with the following function:

```
SCardGetAttrib
  IN SCARDHANDLE hCard,
  IN DWORD dwAttrId,
  OUT LPBYTE pbAttr,
  IN OUT LPDWORD pcbAttrLen );
```

`hCard` is the handle to the card you get from `SCardConnect`.

`dwAttrId` is the identifier for the attribute you want to get. The `dwAttrId` for the ATR is: `SCARD_ATTR_ATR_STRING`.

`pbAttr` is a pointer to a bytearray which contains the data you want from the card (if the function was successful).

`pcbAttrLen` is a pointer to an integer or dword that contains the length of the data you get.

If the function is successful it returns `SCARD_S_SUCCESS`.

1.7 Reading data from a synchronous card

To read the memory of the card you need to use these functions:

- `SCardEstablishContext` to get the context you need for `SCardListReaders` and `SCardConnect`.
- `SCardListReaders` to get the names of all installed readers; one of them is needed for `SCardConnect`.
- `SCardConnect` to open a connection to the card.
- `SCardDisconnect` to close the connection to the card.

To read data from the card you need to know which card you have. If you don't know which card you have, you can read the ATR from the card (with `SCardGetAttrib`, `dwAttrId`: `SCARD_ATTR_ATR_STRING`). You will get a bytearray, where the third value tells you the card type; 2WBP: 0xA2, 3WBP: 0x92, I2C: 0x49.

- If you have a 2W card you need to use `SCard2WBPReadData`.
- If you have a 3W card you need to use `SCard3WBPReadData`.
- If you have an I2C card you need to use `SCardI2CInit` and `SCardI2CReadData`.

First, you must call `SCardEstablishContext`. With the context you get from that function, you can call `SCardListReaders`. You can then choose one of the listed reader names and use that to call `SCardConnect`.

- If you have a 2WBP or 3WBP card, you can now read the memory.
- If you have an I2C card, you must initialize the Synchronous-API accordingly before you can read data. If `SCardI2CInit` returns no error (`OKERR_OK`) you can read data.

Note: I2C cards have many types. If you take one predefined type you should change parameter 2 to NULL, else you should create an `I2C_Card_Parameters` structure and set the members of these structure to the required settings. This structure has the following members:

- `ucNumberOfAddressBytes` (use default: 1)
- `ucPageSize` (use default: 8)
- `ulMemorySize` (use default: 256)

When reading is complete you should close the connection to the card by using `SCardDisconnect`.

1.8 Writing data to a synchronous card

It is very easy to write data to the card. You need to know only the card PIN and the offset at which you want to write the data.

- First you must connect to the card; see *Section 1.7 Reading data from a synchronous card*.
- Before you can write data to the card, for some card types you must present the PIN to the card. You can do this with the function `SCard2WBPPresentPIN` for 2W cards, or `SCard3WBPPresentPIN` for 3W cards. If this function returns `OKERR_OK`, the presented PIN was correct and you can write to the card. If you have an I2C card you do not need to present the PIN.

```
SCard2WBPWriteData  (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulDataLen,
  IN LPBYTE pbData,
  IN ULONG ulAddress );

SCard3WBPWriteData  (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulDataLen,
  IN LPBYTE pbData,
  IN ULONG ulAddress,
  IN BOOL fProtect );

SCardI2CWriteData  (
  IN SCARDHANDLE ulHandleCard,
  BYTE * pbData,
  ULONG ulWriteBufferSize,
  ULONG ulAddress,
  ULONG ulDataLen );
```

- `ulHandleCard` contains the handle to the smart card obtained from `SCardConnect`.
- `ulDataLen` contains the length of the data you want to write.
- `pbData` is a pointer to a bytearray that contains the data you want to write.
- `ulAddress` contains the offset. Offset specifies where `WriteData` should start to write.
- `fProtect` is a boolean variable which is TRUE if the byte should be protected after successful writing, or FALSE if not.
- `ulWriteBufferSize` contains the size of the data buffer `pbData`.

Note: `ulDataLen` + `ulAddress` should not be greater than 256 for 2W cards, 1024 for 3W cards, or the specified length for I2C cards.

1.9 AVR controller based family and synchronous cards

For some synchronous cards, initialization on the AVR controller chip ends with success at 3 V, but later results of read/write operation are unpredictable, ending with either an error or the receipt of fake data. If you encounter any problems with accessing a card over Synchronous-API with an AVR controller based reader, it is recommended to change the voltage selection order of the reader. This configuration forces synchronous card initialization at 5 V. The voltage selection order for AVR controller based readers should be set to 5 V, 3 V, 1.8 V (0x1B).

Please refer to github repository containing sample code. Additional configuration details can be found in the reader's developer guide.

Changing the voltage selection order requires the following steps:

1. Set the voltage sequence by sending the following APDUs:
   Request: FF 70 07 6B 0B A2 09 A1 07 A3 05 A0 03 82 01 1B 00
   Response: BD 00 90 00.
2. Apply settings:
   Request: FF 70 07 6B 09 A2 07 A1 05 A9 03 80 01 00 00
   Response: 9D 00 90 00

Note: If your reader is AVR ROM based, reader steps 1 and 2 need to be repeated at every power up. This configuration is lost after loss of voltage. In future releases, the HID CCID driver will perform proper initialization of synchronous cards and persistent ROM reader configuration.

1.10 Tools

A number of tools are provided for testing purposes.

1.10.1 Check Protected tool

Note: This program only works with SLE4418 and SLE4428 cards.

This program verifies the protection bit of every byte of the card, then shows if the byte is protected (a red + symbol) or not protected (a green - symbol):

```
0FFSET+ - + - - + - + +
0FFSET+ - + - - - + + -
0FFSET+ - + + + - + + -
```

1.10.2 Read Memory tool

This tool allows you to read (and write) the memory of the card. To read the memory, insert the card and click the **Read Memory** button.

- If you insert a 2WBP or a 3WBP card, the data will be read.
- If you insert an I2C card, you must select the type. If you don't know the type, select **Default**. After you read the data you are also able to write it.

To write data to a 2WBP or 3WBP card, you must present the PIN to the card. If the PIN is correct the **Write Memory** button is enabled. Click the button and enter the data. The offset specifies where the data will be written, and must be in hexadecimal format. You can choose if you want to enter the data as hexadecimal values or as normal ASCII values. When you click **OK**, the data is written to the card and the window is updated with the read data.

Note: If you have an I2C card, the **Present Pin** button will be disabled and the **Write Memory** button will be enabled after reading.

1.10.3 Syntst tool

This menu oriented program allows separate testing of each API function. Building this program may cause problems if the original MSVC 6.0 header file **winscard.h** is used. Please use the newer version of the file provided with this API.

1.10.4 Usage of ok.h and other header files from OMNIKEY

The file **ok.h** is intended to be used by all OMNIKEY written code. Its main purpose is to provide a common coding style and help to keep the code operating system independent. It contains type definitions for all common data types and pointers to them. Furthermore, commonly used macros and company-wide error codes are defined here.

There is another include file called **okos.h**, which contains some pragma directives that disable certain unimportant warnings for the Microsoft Compiler. This is necessary to compile **windows.h** under 32-bit at warning level 4. The same pragma directives occur in **ok.h** for the case where **okos.h** is not present. They are properly guarded, to be activated only when MSC is in use. In any C code the include file order should be:

1. Include **okos.h** if necessary (i.e. a `#include <windows.h>` statement will follow).
2. Include the standard compiler files, like **windows.h** and **stdio.h**.
3. Include **ok.h** to get all the OMNIKEY definitions and types.

Section 2

Synchronous-API

This section describes in detail how to start using each card type, and the various functions within the Synchronous-API.

2.1 Using 2WBP cards

For more information about the functions that support smart card services, see MSDN.

2.1.1 Open the connection to the card

First, you must call the function `SCardEstablishContext` to establish the resource manager context:

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

If the function returns `SCARD_S_SUCCESS`, the context has been successfully created. You can now list all installed readers in your computer by using `SCardListReaders`:

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

After you have listed the readers you must select one of them. You can do that through a dialog, or in your application. You can then connect to the card using `SCardConnect`:

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

According to the library implementation, the parameter `dwPreferredProtocols` must be `SCARD_PROTOCOL_T0`. After this function has completed successfully, you are connected to the card. You can now read data, change the PIN, or perform other functions.

2.1.2 Read data from the card

```
OKERR ENTRY SCard2WBPReadData (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulBytesToRead,
  OUT LPBYTE pbData,
  IN ULONG ulAddress );
```

`ulHandleCard` must contain the card handle you obtained from `SCardConnect`.

`ulBytesToRead` sets how many bytes will be read from the card.

`pbData` is a pointer to a bytearray containing the memory read from the card, if the function was successful.

`ulAddress` defines the start offset where the function will start to read.

2.1.3 Close the connection to the card

When you have finished working with the card, you should close the connection using `SCardDisconnect`:

```
LONG SCardDisconnect (
  IN SCARDHANDLE hCard,
  IN DWORD dwDisposition );
```

2.2 SCard2WBP overview

The abbreviation 'SCard2WBP' is used here to refer to the following smart cards from Siemens which are supported by this shared library:

- SLE4432
- SLE4442 (with PIN security logic)

For a full understanding of the operation and functions of these cards, please refer to the Siemens technical manuals:

- Siemens ICs for Chip cards, SLE4432 / SLE4442
- Intelligent 256 Byte EEPROM, Data Sheet 01.94

The SCard2WBP module provides access to the services of the SLE4432/4442 card. The function calls deal with communications with the SLE4432/4442 and the passing of parameters.

Note: Your reader type determines which synchronous smart cards are supported. Please refer to *Section 1.5 Cards supported by the OMNIKEY reader*.

The following functions are available in the SCard2WBP module:

- SCard2WBPReadData
- SCard2WBPReadProtectionMemory
- SCard2WBPWriteData
- SCard2WBPCompareAndProtect
- SCard2WBPPresentPIN
- SCard2WBPChangePIN
- SCard2WBPIsPinPresented

2.2.1 SCard2WBPChangePIN

This function changes the PIN of the card.

```
OKERR ENTRY SCard2WBPChangePIN  (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulOldPINLen,
  IN LPBYTE pbOldPIN,
  IN ULONG ulNewPINLen,
  IN LPBYTE pbNewPIN );
```

The length of the old (current) and the new PIN must be three bytes.

Note: In this API the 'Programmable Security Code' (PSC) is called PIN.

Parameters:

`ulHandleCard`	Handle of the smart card (get from `SCardConnect`).
`ulOldPINLen`	Length of the old PIN (must be 3).
`pbOldPIN`	Pointer to the old PIN.
`ulNewPINLen`	Length of the new PIN (must be 3).
`pbNewPIN`	Pointer to the new PIN.

Returns:

OK Standard Error Codes; see header file **ok.h**.

See also:

SCard2WBPPresentPIN

Attention:

For AVR controller based readers, see *Section 1.9 AVR controller based family and synchronous cards*.

2.2.2 SCard2WBPCompareAndProtect

This function corresponds to the 'Write Protection Memory' operation (refer to the manuals listed in *Section 2.2 SCard2WBP overview*). The execution of this command involves a comparison of the entered data byte `bData` with the assigned byte in the EEPROM (assigned by `ulAddress`). If they match, the protection bit is written, making the data unchangeable. If there is not a match, writing of the protection bit is suppressed.

```
OKERR ENTRY SCard2WBPCompareAndProtect(
  IN SCARDHANDLE ulHandleCard,
  IN BYTE bData,
  IN ULONG ulAddress );
```

Only the first 32 bytes (0...31) of the main memory can be protected.

Parameters:

`ulHandleCard`	Handle of the smart card (get from `SCardConnect`).
`bData`	Byte to be matched.
`ulAddress`	Memory address (0...31).

Returns:

OK Standard Error Codes; see header file **ok.h**.

See also:

SCard2WBPReadProtectionMemory

Attention:

For AVR controller based readers, see *Section 1.9 AVR controller based family and synchronous cards*.

2.2.3 SCard2WBPIsPinPresented

```
OKERR ENTRY SCard2WBPIsPinPresented  (
  IN SCARDHANDLE ulHandleCard,
  OUT LPBOOL pfPinPresented );
```

Parameters:

`ulHandleCard`	Handle of the smart card (get from `SCardConnect`)
`pfPinPresented`	TRUE if PIN is already presented.

Returns:

OK Standard Error Codes; see header file **ok.h**.

See also:

SCard2WBPPresentPIN

Attention:

For AVR controller based readers, see *Section 1.9 AVR controller based family and synchronous cards*.

2.2.4 SCard2WBPPresentPIN

This function sends the PIN `pbPin` to the card, which is followed by some commands to check the PIN.

```
OKERR ENTRY SCard2WBPPresentPIN  (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulPINLen,
  IN LPBYTE pbPIN );
```

Note: The SLE4442 requires a correct verification of the 'Programmable Security Code' (PSC) stored in the Security Memory for altering data. In this API the 'Programmable Security Code' (PSC) is called PIN.

Parameters:

`ulHandleCard`	Handle of the smart card (get from `SCardConnect`).
`ulPINLen`	Length of the PIN (must be 3).
`pbPIN`	Pointer to the PIN.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_OK` if PIN is correct.

`OKERR_PW_WRONG` if PIN is incorrect.

See also:

SCard2WBPChangePIN

Attention:

For AVR controller based readers, see *Section 1.9 AVR controller based family and synchronous cards*.

2.2.5 SCard2WBPReadData

This function corresponds to the 'Read Main Memory' operation (refer to the manuals listed in *Section 2.2 SCard2WBP overview*) and reads one or multiple bytes from the card. The function reads out the content of the main memory (LSB first) starting at the given address (`ulAddress` = 0...255). Read access to the main memory is always possible.

```
OKERR ENTRY SCard2WBPReadData  (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulBytesToRead,
  OUT LPBYTE pbData,
  IN ULONG ulAddress );
```

Parameters:

`ulHandleCard`	Handle of the smart card (get from `SCardConnect`).
`ulBytesToRead`	Length of the data buffer.
`pbData`	Pointer to the data buffer.
`ulAddress`	Start offset for the read operation (0...255).

Returns:

OK Standard Error Codes; see header file **ok.h**.

See also:

SCard2WBPWriteData

Attention:

For AVR controller based readers, see *Section 1.9 AVR controller based family and synchronous cards*.

2.2.6 SCard2WBPReadErrorCounter

This function reads the error counter from the 'Security Memory'.

```
OKERR ENTRY SCard2WBPReadErrorCounter  (
  IN SCARDHANDLE ulHandleCard,
  OUT LPBYTE pbCounter );
```

Parameters:

`ulHandleCard`	Handle of the smart card (get from `SCardConnect`).
`pbCounter`	Pointer to the error counter.

Returns:

OK Standard Error Codes; see header file **ok.h**.

Attention:

For AVR controller based readers, see *Section 1.9 AVR controller based family and synchronous cards*.

2.2.7 SCard2WBPReadProtectionMemory

This function corresponds to the 'Read Protection Memory' operation (refer to the manuals listed in *Section 2.2 SCard2WBP overview*) and reads the protection bits of the first 32 bytes (byte 0 - byte 31).

```
OKERR ENTRY SCard2WBPReadProtectionMemory  (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulDataLen,
  OUT LPBYTE pbData );
```

Note: The length of `ulDataLen` must be four bytes.

Please note that the bit order of each byte is reversed, as shown in the following table:

Protection memory	Returned bytes	
Byte 0	Bit 7	Byte 1
Byte 1	Bit 6	
...	...	
Byte 6	Bit 1	
Byte 7	Bit 0	
Byte 8	Bit 7	Byte 2
Byte 9	Bit 6	
...	...	
Byte 14	Bit 1	
Byte 15	Bit 0	
Byte 16	Bit 7	Byte 3
Byte 17	Bit 6	
...	...	
Byte 22	Bit 1	
Byte 23	Bit 0	
Byte 24	Bit 7	Byte 4
Byte 25	Bit 6	
...	...	
Byte 30	Bit 1	
Byte 31	Bit 0	

Parameters:

`ulHandleCard`	Handle of the smart card (get from `SCardConnect`).
`ulDataLen`	Length of the data buffer (must be 4).
`pbData`	Pointer to the data buffer.

Returns:

OK Standard Error Codes; see header file **ok.h**.

See also:

SCard2WBPCompareAndProtect

Attention:

For AVR controller based readers, see *Section 1.9 AVR controller based family and synchronous cards*.

2.2.8 SCard2WBPWriteData

This function corresponds to the 'Update Main Memory' operation (refer to the manuals listed in *Section 2.2 SCard2WBP overview*) and writes one or multiple bytes to the card. The function writes the data in the main memory starting at the given address (`ulAddress` = 0...255).

```
OKERR ENTRY SCard2WBPWriteData  (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulDataLen,
  IN LPBYTE pbData,
  IN ULONG ulAddress );
```

Note: The SLE4442 requires a correct presentation of the PIN (function `SCard2WBPPresentPIN` must have been called) for altering main memory data.

Parameters:

`ulHandleCard`	Handle of the smart card (get from `SCardConnect`).
`ulDataLen`	Length of the data buffer.
`pbData`	Pointer to the data buffer.
`ulAddress`	Start offset for the write operation (0...255).

Returns:

OK Standard Error Codes; see header file **ok.h**.

See also:

SCard2WBPReadData

SCard2WBPPresentPIN

Attention:

For AVR controller based readers, see *Section 1.9 AVR controller based family and synchronous cards*.

2.3 Using 3WBP cards

For more information about the functions that support smart card services, see MSDN.

2.3.1 Open the connection to the card

First, you must call the function `SCardEstablishContext` to establish the resource manager context:

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

If the function returns `SCARD_S_SUCCESS`, the context has been successfully created. You can now list all installed readers in your computer by using `SCardListReaders`:

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

After you have listed the readers you must select one of them. You can do that through a dialog, or in your application. You can then connect to the card using `SCardConnect`:

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

According to the library implementation, the parameter `dwPreferredProtocols` must be `SCARD_PROTOCOL_T0`. After this function has completed successfully, you are connected to the card. You can now read the data, change the PIN, or perform other functions.

2.3.2 Read data from the card

```
OKERR ENTRY SCard3WBPReadData (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulBytesToRead,
  OUT LPBYTE pbData,
  IN ULONG ulAddress );
```

`ulHandleCard` must contain the card handle you obtained from `SCardConnect`.
`ulBytesToRead` sets how many bytes will be read from the card.
`pbData` is a pointer to a bytearray containing the memory read from the card, if the function was successful.
`ulAddress` defines the start offset where the function will start to read.

2.3.3 Close the connection to the card

When you have finished working with the card, you should close the connection using `SCardDisconnect`:

```
LONG SCardDisconnect (
  IN SCARDHANDLE hCard,
  IN DWORD dwDisposition );
```

2.4 SCard3WBP overview

The abbreviation SCard3WBP is used here to refer to the following smart cards from Siemens which are supported by this shared library:

- SLE4418
- SLE4428 (with PIN security logic)

For a full understanding of the operation and functions of these cards, please refer to the Siemens technical manuals:

- Siemens ICs for Chip cards
- SLE4418 / SLE4428 intelligent 8-Kbit EEPROM
- Data Sheet 09.92

The SCard3WBP module provides access to the services of the SLE4418/4428 card. The function calls deal with communications with the SLE4418/4428 and passing of the parameters.

Note: Your reader type determines which synchronous smart cards are supported. Please refer to *Section 1.5 Cards supported by the OMNIKEY reader*.

The following functions are available in the SCard3WBP module:

- SCard3WBPReadData
- SCard3WBPVerifyProtectBit
- SCard3WBPWriteData
- SCard3WBPCompareAndProtect
- SCard3WBPPresentPIN
- SCard3WBPChangePIN
- SCard3WBPIsPinPresented
- SCard3WBPVerifyProtectBitEx

2.4.1 SCard3WBPChangePIN

This function changes the PIN of the card. The length of the old (current) and the new PIN must be two bytes.

```
OKERR ENTRY SCard3WBPChangePIN  (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulOldPINLen,
  IN LPBYTE pbOldPIN,
  IN ULONG ulNewPINLen,
  IN LPBYTE pbNewPIN );
```

Parameters:

`ulHandleCard`	Handle of the smart card (get from `SCardConnect`).
`ulOldPINLen`	Length of the old PIN (must be 2).
`pbOldPIN`	Pointer to the old PIN.
`ulNewPINLen`	Length of the new PIN (must be 2).
`pbNewPIN`	Pointer to the new PIN.

Returns:

OK Standard Error Codes; see header file **ok.h**.

See also:

SCard3WBPPresentPIN

Attention:

For AVR controller based readers, see *Section 1.9 AVR controller based family and synchronous cards*.

2.4.2 SCard3WBPCompareAndProtect

This function corresponds to the 'Write protect bit with data comparison' operation (refer to the manuals listed in *Section 2.4 SCard3WBP overview*) to set a byte 'read only'. This function is used to compare the byte `bData` and a byte in the card referenced by `ulAddress`. If they match, the byte is write-protected.

```
OKERR ENTRY SCard3WBPCompareAndProtect  (
  IN SCARDHANDLE ulHandleCard,
  IN BYTE bData,
  IN ULONG ulAddress );
```

Parameters:

`ulHandleCard`	Handle of the smart card (get from `SCardConnect`).
`bData`	Byte to be matched.
`ulAddress`	Memory address.

Returns:

OK Standard Error Codes; see header file **ok.h**.

See also:

SCard3WBPWriteData

Attention:

For AVR controller based readers, see *Section 1.9 AVR controller based family and synchronous cards*.

2.4.3 SCard3WBPIsPinPresented

```
OKERR ENTRY SCard3WBPIsPinPresented  (
  IN SCARDHANDLE ulHandleCard,
  OUT LPBOOL pfPinPresented );
```

Parameters:

`ulHandleCard`	Handle of the smart card (get from `SCardConnect`).
`pfPinPresented`	TRUE if PIN is already presented.

Returns:

OK Standard Error Codes; see header file **ok.h**.

See also:

SCard3WBPPresentPIN

Attention:

For AVR controller based readers, see *Section 1.9 AVR controller based family and synchronous cards*.

2.4.4 SCard3WBPPresentPIN

This function uses the 'Write error counter', the Compare 1st PIN, and the 2nd PIN operation (refer to the manuals listed in *Section 2.4 SCard3WBP overview*) to unlock the card. This function is used to enable erasing and writing of the card. The PIN is referenced by `pbPIN` and must be two bytes long, so `ulPinLen` must also be two.

```
OKERR ENTRY SCard3WBPPresentPIN  (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulPINLen,
  IN LPBYTE pbPIN );
```

Note: The function sets the error counter to the maximum value of 8 retries after enabling. Refer to the manuals listed in *Section 2.4 SCard3WBP overview* for details on PIN, transport code and error counter.

Parameters:

`ulHandleCard`	Handle of the smart card (get from `SCardConnect`).
`ulPINLen`	Length of the PIN (must be 2).
`pbPIN`	Pointer to the PIN.

Returns:

OK Standard Error Codes; see header file **ok.h**.

See also:

SCard3WBPChangePIN

SCard3WBPWriteData

Attention:

For AVR controller based readers, see *Section 1.9 AVR controller based family and synchronous cards*.

2.4.5 SCard3WBPReadData

This function corresponds to the 'Read 8 bits data without protect bit' operation (refer to the manuals listed in *Section 2.4 SCard3WBP overview*) and reads one or multiple bytes from the card. This function is used to read a block of data of up to 1024 bytes (the maximum storage capacity) from the card into a buffer referenced by `pbData`. `ulAddress` specifies the start address for the read operation on the card.

```
OKERR ENTRY SCard3WBPReadData  (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulBytesToRead,
  OUT LPBYTE pbReadBuffer,
  IN ULONG ulAddress );
```

Note: `ulAddress` + `ulBytesToRead` must not be greater than 1024 (i.e. there is no wrap around).

Parameters:

`ulHandleCard`	Handle of the smart card (get from `SCardConnect`).
`ulBytesToRead`	Number of bytes to read.
`pbReadBuffer`	Pointer to the buffer.
`ulAddress`	Start offset for read operation.

Returns:

OK Standard Error Codes; see header file **ok.h**.

See also:

SCard3WBPWriteData

Attention:

For AVR controller based readers, see *Section 1.9 AVR controller based family and synchronous cards*.

2.4.6 SCard3WBPVerifyProtectBit

This function uses the 'Read 9 bits data with protect bit' operation (refer to the manuals listed in *Section 2.4 SCard3WBP overview*) to determine if a byte is 'read only'. This function is used to check a byte in the card referenced by `ulAddress`. The flag referenced by `pfProtect` holds TRUE if the byte has the write protection set.

```
OKERR ENTRY SCard3WBPVerifyProtectBit  (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulAddress,
  OUT LPBOOL pfProtected );
```

Parameters:

`ulHandleCard`	Handle of the smart card (get from `SCardConnect`).
`ulAddress`	Memory address.
`pfProtected`	Pointer to the result.

Returns:

OK Standard Error Codes; see header file **ok.h**.

See also:

SCard3WBPWriteData

SCard3WBPCompareAndProtect

Attention:

For AVR controller based readers, see *Section 1.9 AVR controller based family and synchronous cards*.

2.4.7 SCard3WBPVerifyProtectBitEx

This function uses the 'Read 9 bits data with protect bit' operation (refer to the manuals listed in *Section 2.4 SCard3WBP overview*) to determine if a byte is 'read only'. This function is used to check the protection bits of the card starting at `ulAddress`. The values of the protection bits (TRUE or FALSE) are stored in the passed buffer.

```
OKERR ENTRY SCard3WBPVerifyProtectBitEx  (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulBytesToRead,
  OUT LPBYTE pbReadBuffer,
  IN ULONG ulAddress );
```

Note: If the protection bits of 10 bytes are checked, the buffer must have a size of at least 10 bytes.

The protection bit of the first byte is stored in the first byte of the buffer, the protection bit of the second byte is stored in the second byte of the buffer, etc.

Parameters:

`ulHandleCard`	Handle of the smart card (get from `SCardConnect`).
`ulBytesToRead`	Number of bytes to check.
`pbReadBuffer`	Pointer to the buffer.
`ulAddress`	Start offset for check operation.

Returns:

OK Standard Error Codes; see header file **ok.h**.

See also:

SCard3WBPWriteData

SCard3WBPCompareAndProtect

Attention:

For AVR controller based readers, see *Section 1.9 AVR controller based family and synchronous cards*.

2.4.8 SCard3WBPWriteData

This function uses either the 'Erase and write without protect bit' operation or 'Erase and write with protect bit' operation to write one or multiple bytes to the card. This is used to write a block of data of up to 1024 bytes (the maximum storage capacity) to the card. `ulAddress` specifies the start address for the write operation on the card.

```
OKERR ENTRY SCard3WBPWriteData  (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulDataLen,
  IN LPBYTE pbData,
  IN ULONG ulAddress,
  IN BOOL fProtect );
```

Note: `ulAddress` + `ulDataLen` must not be greater than 1024 (e.g. there is no wrap around).

`fProtect = TRUE` will set the write protection for every byte written. The PIN must have been presented successfully before writing is possible on the SLE4428 card. After writing the block of data, the function performs an internal read-back and compare operation to verify that every byte has been written correctly.

Parameters:

`ulHandleCard`	Handle of the smart card.
`ulDataLen`	Length of the data buffer.
`pbData`	Pointer to the data buffer.
`ulAddress`	Start offset of write operation.
`fProtect`	Write and if 'TRUE' set protect bit.

Returns:

OK Standard Error Codes; see header file **ok.h**.

See also:

SCard3WBPReadData

SCard3WBPPresentPIN

Attention:

For AVR controller based readers, see *Section 1.9 AVR controller based family and synchronous cards*.

2.5 Using I2C cards

For more information about the functions that support smart card services, see MSDN.

2.5.1 Open the connection to the card

First, you must call the function `SCardEstablishContext` to establish the resource manager context:

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

If the function returns `SCARD_S_SUCCESS`, the context has been successfully created. You can now list all installed readers in your computer by using `SCardListReaders`:

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

After you have listed the readers you must select one of them. You can do that through a dialog, or in your application. You can then connect to the card using `SCardConnect`:

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

According to the library implementation, the parameter `dwPreferredProtocols` must be `SCARD_PROTOCOL_T0`. After this function has completed successfully, you are connected to the card. You must now initialize the card to perform other functions.

2.5.2 Initialize the card

```
OKERR ENTRY SCardI2CInit  (
  IN SCARDHANDLE ulHandleCard,
  IN SCARD_I2C_CARD_PARAMETERS * pCardParameters,
  IN SCARD_I2C_TYPE Type );
```

`ulHandleCard` must contain the card handle you obtained from the `SCardConnect` function.

`pCardParameters` is a pointer to a `SCARD_I2C_CARD_PARAMETERS` structure which contains these members:

- `ucNumberOfAddressBytes` (default: 1)
- `ucPageSize` (default: 8)
- `ulMemorySize` (default: 256)

Note: A `ucPageSize` of 0 is a special case that indicates a page size of 256.

You can now read or write data.

2.5.3 Read data from the card

```
OKERR ENTRY SCardI2CReadData (
  IN SCARDHANDLE ulHandleCard,
  BYTE * pbReadBuffer,
  ULONG ulReadBufferSize,
  ULONG ulAddress,
  ULONG ulBytesToRead );
```

`ulHandleCard` must contain the card handle you obtained from `SCardConnect`.

`ulBytesToRead` sets how many bytes will be read from the card.

`ulAddress` defines the start offset where the function will start to read.

`ulReadBufferSize` contains the size of `pbReadBuffer`.

`pbReadBuffer` must be a pointer to a bytearray that contains the memory read from the card, if the function was successful.

2.5.4 Close the connection to the card

When you have finished working with the card, you should close the connection using `SCardDisconnect`:

```
LONG SCardDisconnect (
  IN SCARDHANDLE hCard,
  IN DWORD dwDisposition );
```

2.6 SCardI2C overview

The following I2C bus cards are supported by **scardsyn.dll** shared library:

- I2C cards from STMicroelectronics: ST14C02C ST14C04C ST14E32 M14C04 M14C16 M14C32 M14C64 M14128 M14256
- I2C cards from GEMplus (Gemalto): GFM2K GFM4K GFM32K
- I2C cards from Atmel: AT24C01A AT24C02 AT24C04 AT24C08 AT24C16 AT24C164 AT24C32 AT24C64 AT24C128 AT24C256 AT24CS128 AT24CS256 AT24C512 AT24C1024

For a full understanding of the operation and functions of these cards, please refer to the corresponding I2C bus card technical manual. If the I2C bus card you intend to use is not among the predefined cards above, you must allocate and initialize a card parameters structure and submit its address when calling the `SCardI2CInit` function.

Note: Your reader type determines which synchronous smart cards are supported. Please refer to *Section 1.5 Cards supported by the OMNIKEY reader*.

The following functions are available in the SCardI2C module included in this library:

- SCardI2CInit
- SCardI2CReadData
- SCardI2CWriteData

2.6.1 SCardI2CInit

This function initializes the card and protocol specific parameters of the driver for communication with the I2C bus card. It must be called once before any use of `SCardI2CReadData` or `SCardI2CWriteData`.

```
OKERR ENTRY SCardI2CInit  (
  IN SCARDHANDLE ulHandleCard,
  IN SCARD_I2C_CARD_PARAMETERS * pCardParameters,
  IN SCARD_I2C_TYPE Type );
```

There is no corresponding function in the card itself. The card is specified with `Type`. In this case, the card parameters are internally initialized according to the corresponding manufacturer specification, and the `pCardParameters` pointer is not evaluated. When `Type = NO_PREDEFINED_CARD_PARAMETERS`, each card parameter is defined in a `CardParameters` structure which must be allocated and initialized by the calling application. Its address is submitted as the `pCardParameters` pointer in the call of this function.

Note: It is expected that the card is already connected.

The following `Type` constants can be used:

- STMicroelectronics: ST14C02C ST14C04C ST14E32 M14C04 M14C16 M14C32 M14C64 M14128 M14256
- GEMplus (Gemalto): GFM2K GFM4K GFM32K
- Atmel: AT24C01A AT24C02 AT24C04 AT24C08 AT24C16 AT24C164 AT24C32 AT24C64 AT24C128 AT24C256 AT24CS128 AT24CS256 AT24C512 AT241024

Parameters:

`ulHandleCard`	Handle to a I2C bus card, provided from the “smart card resource manager” after connecting to the card (`SCardConnect`).
`pCardParameters`	Pointer to a structure holding I2C card parameters. Used only if `Type = NO_PREDEFINED_CARD_PARAMETERS`.
`Type`	Predefined type of the I2C card.

Returns:

OK Standard Error Codes; see header file **ok.h**.

Attention:

For AVR controller based readers, see *Section 1.9 AVR controller based family and synchronous cards*.

2.6.2 SCardI2CReadData

Reads a number of bytes, starting from the specified memory address `ulAddress`, and saves the content in the read buffer pointed to by `pbReadBuffer`. The function uses internal I2C bus sequential read for the number of bytes. Better performance is achieved when a block of bytes is read with one call of this function, instead of calling it in a loop reading 1 byte at a time.

```
OKERR ENTRY SCardI2CReadData  (
  IN SCARDHANDLE ulHandleCard,
  BYTE * pbReadBuffer,
  ULONG ulReadBufferSize,
  ULONG ulAddress,
  ULONG ulBytesToRead );
```

Parameters:

`ulHandleCard`	Handle (get from `SCardConnect`).
`pbReadBuffer`	Pointer to the buffer where the data read from the card are to be stored.
`ulReadBufferSize`	Size of the read buffer.
`ulAddress`	Memory address to read from.
`ulBytesToRead`	Number of bytes to be read from the `ulAddress`.

Returns:

OK Standard Error Codes; see header file **ok.h**.

Attention:

For AVR controller based readers, see *Section 1.9 AVR controller based family and synchronous cards*.

2.6.3 SCardI2CWriteData

Writes a number of bytes (`ulBytesToWrite`) to the specified card memory, starting from `ulAddress`. The data to be written is taken from the buffer pointed to by `pbWriteBuffer`. The function uses internal I2C bus sequential write for the number of bytes. Better performance is achieved when a block of bytes is written with one call of this function, instead of calling it in a loop writing 1 byte at a time.

```
OKERR ENTRY SCardI2CWriteData  (
  IN SCARDHANDLE ulHandleCard,
  BYTE * pbWriteBuffer,
  ULONG ulWriteBufferSize,
  ULONG ulAddress,
  ULONG ulBytesToWrite );
```

Parameters:

`ulHandleCard`	Handle of the smart card (get from `SCardConnect`).
`pbWriteBuffer`	Pointer to the write buffer holding the data to be written.
`ulWriteBufferSize`	Size of the write buffer.
`ulAddress`	Memory address in the I2C card where writing to the buffer will start.
`ulBytesToWrite`	Number of bytes to be written in the `ulAddress`.

Returns:

OK Standard Error Codes; see header file **ok.h**.

Attention:

For AVR controller based readers, see *Section 1.9 AVR controller based family and synchronous cards*.

2.7 Using SCard4404 cards

The abbreviation SCard4404 is used here to refer to the following smart card from Siemens, which are supported by this shared library:

- SLE4404

For a full understanding of the operation and functions of this card, please refer to the Siemens technical manual:

- Siemens ICs for Chip cards, SLE4404
- Intelligent 416 Bit EEPROM with Internal PIN Check Data Sheet 04.94

The SCard4404 module provides access to the services of the SLE4404 card. The function calls deal with communications with the SLE4404.

The following functions are available in the SCard4404 module:

- SCard4404ReadData
- SCard4404WriteData
- SCard4404EraseErrorCounter
- SCard4404EraseScratchPadMemory
- SCard4404EraseUserMemory

2.7.1 SCard4404ChangeUserCode

This function exchanges the old user code with the new one. This is done by verifying the old user code first, erasing the user code area, and then writing the new user code. Finally a verification of the new user code is performed.

```
OKERR ENTRY SCard4404ChangeUserCode  (
  IN SCARDHANDLE ulHandleCard,
  IN LPBYTE pbOldPin,
  IN LPBYTE pbNewPin );
```

Parameters:

`ulHandleCard`	Handle of the smart card.
`pbOldPin`	Pointer to the data buffer.
`pbNewPin`	Pointer to the data buffer.

Returns:

OK Standard Error Codes; see header file **ok.h**.

See also:

SCard4404WriteData

SCard4404VerifyUserCode

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.7.2 SCard4404EraseErrorCounter

This function performs the erase operation on the error counter.

```
OKERR ENTRY SCard4404EraseErrorCounter  (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulBitAddress );
```

Note: Correct verification of the user code is mandatory before calling this function.

Parameters:

`ulHandleCard`	Handle of the smart card.
`ulBitAddress`	Address of any bit in the error counter.

Returns:

OK Standard Error Codes; see header file **ok.h**.

See also:

SCard4404EraseScratchPadMemory

SCard4404EraseUserMemory

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.7.3 SCard4404EraseScratchPadMemory

This function performs the erase operation on the scratch pad memory.

```
OKERR ENTRY SCard4404EraseScratchPadMemory  (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulBitAddress );
```

Note: Correct verification of the user code is mandatory before calling this function.

Parameters:

`ulHandleCard`	Handle of the smart card.
`ulBitAddress`	Address of any bit in the scratch pad memory.

Returns:

OK Standard Error Codes; see header file **ok.h**.

See also:

SCard4404EraseErrorCounter

SCard4404EraseUserMemory

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.7.4 SCard4404EraseUserMemory

This function performs the erase operation on the user memory. It writes a non-written bit in the memory counter area when `ulBitAddress` is set to 0, immediately followed by an erase-operation without incrementing the address.

```
OKERR ENTRY SCard4404EraseUserMemory  (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulBitAddress );
```

Note: Correct verification of user code and / or memory code is mandatory before calling this function.

Parameters:

`ulHandleCard`	Handle of the smart card.
`ulBitAddress`	Address of any non-written bit of the memory counter.

Returns:

OK Standard Error Codes; see header file **ok.h**.

See also:

SCard4404EraseErrorCounter

SCard4404EraseScratchPadMemory

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.7.5 SCard4404ReadData

This function reads the number of bits (`ulBitsToRead`) from the specified bit-address (`ulBitAddress`). The data read is stored in the buffer pointed to by `pbReadBuffer`.

```
OKERR ENTRY SCard4404ReadData  (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulBitsToRead,
  IN ULONG ulBitAddress,
  OUT LPBYTE pbReadBuffer,
  ULONG ulReadBufferSize );
```

Note: For non-readable memory address (user code, memory code, read-protected user memory) the bit value returned by the SLE4404 is always 1.

Parameters:

`ulHandleCard`	Handle of the smart card.
`ulBitsToRead`	Length of the data buffer.
`ulBitAddress`	Start offset for the read operation.
`pbReadBuffer`	Pointer to the data buffer.
`ulReadBufferSize`	Pointer to the data buffer.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`pbReadBuffer`: A pointer to the data buffer.

See also:

SCard4404WriteData

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.7.6 SCard4404VerifyUserCode

This function performs the entry of the user code (refer to the manuals listed in *Section 2.7 Using SCard4404 cards*). It presents the user code. It searches for a “1” bit in the error counter and writes the bit to “0”. Finally it will erase the error counter. If all bits of the error counter are erased (=“1”) the verification was successful.

```
OKERR ENTRY SCard4404VerifyUserCode  (
  IN SCARDHANDLE ulHandleCard,
  IN LPBYTE pbData );
```

Parameters:

`ulHandleCard`	Handle of the smart card.
`pbData`	Pointer to the data buffer.

Returns:

OK Standard Error Codes; see header file **ok.h**.

See also:

SCard4404WriteData

SCard4404ReadData

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.7.7 SCard4404WriteData

This function writes data to the card. Writing on a SLE4404 card means clearing bits, i.e. it is only possible to change bits from one to zero, but not vice-versa. If there is a write request where an already cleared bit should be set to 1, an error will be returned and the memory area must be erased first.

```
OKERR ENTRY SCard4404WriteData  (
  IN SCARDHANDLE ulHandleCard,
  IN LPBYTE pbData,
  IN ULONG ulBitsToWrite,
  IN ULONG ulBitAddress );
```

Note: Correct verification of user code and / or memory code is mandatory before calling this function.

Parameters:

`ulHandleCard`	Handle of the smart card.
`pbData`	Pointer to the data buffer.
`ulBitsToWrite`	Number of bits to write.
`ulBitAddress`	Start offset for the write operation.

Returns:

OK Standard Error Codes; see header file **ok.h**.

See also:

SCard4404ReadData

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.8 Using DataFlash cards

For more information about the functions that support smart card services, see MSDN.

2.8.1 Open the connection to the card

This section describes the functions which must be called before the **scardsyn.dll** shared library can be used. First, you must call the function `SCardEstablishContext` to establish the resource manager context:

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

If the function returns `SCARD_S_SUCCESS`, the context has been successfully created. You can now list all installed readers in your computer by using `CardListReaders`.

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

After you have listed the readers you must select one of them. You can do that through a dialog, or in your application. You can then connect to the card using `SCardConnect`:

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

According to the library implementation, the parameter `dwPreferredProtocols` must be `SCARD_PROTOCOL_T0`. After a successful connection to the card, the `SCardDfInit` Synchronous-API function can be called. It initializes library internal data structures according to the `Data_Flash_Type` parameter. Please check if the card type is supported.

```
OKERR ENTRY CardDfInit  (
  IN SCARDHANDLE ulHandleCard,
  IN SCARD_DATAFLASH_TYPE Type );
```

`ulHandleCard` must contain the card handle you obtained through the pointer `phCard` after calling `SCardConnect`. This handle identifies the connection to the DataFlash card in the designated reader.

`Type` is a predefined constant; see *Section 2.9 SCardDF overview*. Only AT45DB041B is currently supported.

After the `SCardDfInit` function has completed successfully, you can use the rest of the **scardsyn.dll** library `SCardDF`... functions.

2.8.2 Close the connection to the card

When you have finished working with the card, you should close the connection using `SCardDisconnect`.

```
LONG SCardDisconnect  (
  IN SCARDHANDLE ulHandleCard,
  IN DWORD dwDisposition );
```

2.9 SCardDF overview

Currently the **scardsyn.dll** shared library supports only the following DataFlash card from ATMEL:

- AT45DB041

The constant AT45DB041 must be used as the `Type` parameter in the call to `SCardDfInit`.

For a full understanding of the operation and functions of these cards, please refer to the corresponding DataFlash card technical manual.

The following functions are available in the **SCardDf** module included in this library:

- SCardDfInit
- SCardDfContinuousArrayRead
- SCardDfMainMemoryPageRead
- SCardDfBufferRead
- SCardDfStatusRegisterRead
- SCardDfBufferWrite
- SCardDfBufferToMainMemoryPageProgram
- SCardDfPageErase
- SCardDfBlockErase
- SCardDfMainMemoryPageProgramThroughBuffer
- SCardDfMainMemoryPageToBufferTransfer
- SCardDfMainMemoryPageToBufferCompare
- SCardDfAutoPageRewriteThroughBuffer

2.9.1 SCardDfAutoPageRewriteThroughBuffer

This mode is needed only if multiple bytes within a page or multiple pages of data are modified in a random fashion. This mode is a combination of two operations: Main Memory Page to Buffer Transfer, and Buffer to Main Memory Page Program with Built-In Erase. A page of data is first transferred from the main memory to buffer 1 or buffer 2, and then the same data is programmed back into its original page of main memory.

```
OKERR ENTRY SCardDfAutoPageRewriteThroughBuffer  (
  IN SCARDHANDLE ulHandleCard,
  IN UCHAR ucBufferNumber,
  IN ULONG ulPageAdr );
```

Parameters:

`ulHandleCard`	Handle to the DataFlash card.
`ucBufferNumber`	Buffer number 1 or 2 to rewrite.
`ulPageAdr`	Number of the page in the main memory to be rewritten.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized or DataFlash card type is not supported.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.9.2 SCardDfBlockErase

A block of pages can be erased with this single operation. Erases the block specified with `ulBlockNr` number in the main memory.

```
OKERR ENTRY SCardDfBlockErase  (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulBlockNr );
```

Parameters:

`ulHandleCard`	Handle to the DataFlash card.
`ulBlockNr`	Number of the block to be erased.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized or DataFlash card type is not supported.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.9.3 SCardDfBufferRead

Data can be read from either of the two buffers, indicated by `ucBufferNumber`. For the DataFlash cards supported there are two buffers, numbered 1 and 2. When the end of a buffer is reached, the read will continue at the beginning of the buffer. The application must provide a read buffer with the necessary length.

```
OKERR ENTRY SCardDfBufferRead  (
  IN SCARDHANDLE ulHandleCard,
  IN BYTE * pbReadBuffer,
  IN ULONG ulReadBufferSize,
  IN ULONG ulByteAdr,
  IN ULONG ulBytesToRead,
  IN UCHAR ucBufferNumber,
  OUT PULONG pulBytesReturned );
```

Parameters:

`ulHandleCard`	Handle to the DataFlash card.
`pbReadBuffer`	Pointer to the buffer where the data read will be stored.
`ulReadBufferSize`	Read buffer size.
`ulByteAdr`	Byte number of the start address in the buffer.
`ulBytesToRead`	Number of bytes to read.
`ucBufferNumber`	Number 1 or 2 of the buffer to read from.
`pulBytesReturned`	Number of bytes actually read.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized or DataFlash card type is not supported

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.9.4 SCardDfBufferToMainMemoryPageProgram

Data written into either buffer 1 or buffer 2 can be programmed into main memory with or without initial erasure of the destination page. The `fErase` parameter specifies if the destination page in the main memory is to be erased first. Successive page programming operations without performing a page erase are not recommended.

```
OKERR ENTRY SCardDfBufferToMainMemoryPageProgram  (
  IN SCARDHANDLE ulHandleCard,
  IN UCHAR ucBufferNumber,
  IN ULONG ulPageAdr,
  IN BOOL fErase );
```

Parameters:

`ulHandleCard`	Handle to the DataFlash card.
`ucBufferNumber`	Number 1 or 2 of the buffer to transfer.
`ulPageAdr`	Destination page number to be programmed.
`fErase`	0 = program without erase. 1 = program with erase.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized or DataFlash card type is not supported.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.9.5 SCardDfBufferWrite

Writes the data pointed to by `pbWriteBuffer` into the DataFlash buffer indicated by `ucBufferNumber`. If the end of the buffer is reached, the write will wrap back to the beginning of the buffer.

```
OKERR ENTRY SCardDfBufferWrite  (
  IN SCARDHANDLE ulHandleCard,
  IN BYTE * pbWriteBuffer,
  IN ULONG ulAddress,
  IN ULONG ulBytesToWrite,
  IN UCHAR ucBufferNumber );
```

Parameters:

`ulHandleCard`	Handle to the DataFlash card.
`pbWriteBuffer`	Pointer to the buffer holding data to be written.
`ulByteAdr`	Byte number of the start address in the buffer.
`ulBytesToWrite`	Number of bytes to write.
`ucBufferNumber`	Number 1 or 2 of the buffer to be written.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized or DataFlash card type is not supported.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.9.6 SCardDfContinuousArrayRead

Sequentially reads continuous number of bytes from the main memory. Depending on the start byte address `ulByteAdr` and `ulBytesToRead`, page boundaries can be crossed. The application must provide a read buffer with the necessary length.

```
OKERR ENTRY SCardDfContinuousArrayRead  (
  IN SCARDHANDLE ulHandleCard,
  IN BYTE * pbReadBuffer,
  IN ULONG ulReadBufferSize,
  IN ULONG ulPageAdr,
  IN ULONG ulByteAdr,
  IN ULONG ulBytesToRead,
  OUT PULONG pulBytesReturned );
```

Parameters:

`ulHandleCard`	Handle to the DataFlash card.
`pbReadBuffer`	Pointer to the buffer where the data read will be stored.
`ulReadBufferSize`	Read buffer size.
`ulPageAdr`	Page number of the start address.
`ulByteAdr`	Page relative byte number of the start address.
`ulBytesToRead`	Number of bytes to read.
`pulBytesReturned`	Number of bytes actually read.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized or DataFlash card type is not supported.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.9.7 SCardDfInit

Initializes the library for the DataFlash card specified with `Type`. The function must be called once at the beginning, before using any other command.

```
OKERR ENTRY SCardDfInit  (
  IN SCARDHANDLE ulHandleCard,
  IN SCARD_DATA_FLASH_TYPE Type );
```

Parameters:

`ulHandleCard`	Handle to the DataFlash card.
`Type`	Type of the DataFlash.

Returns:

`NO_ERROR`: The shared library is initialized.

`OKERR_INIT`: The shared library is not initialized because DataFlash card type is not supported.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.9.8 SCardDfMainMemoryPageProgramThroughBuffer

This operation is a combination of the “buffer write” and “buffer to main memory page program with built in erase” operations. Data is first written in buffer 1 or buffer 2 and then programmed into the specified page in the main memory. If the end of the buffer is reached, the write will wrap to the beginning of the buffer.

```
OKERR ENTRY SCardDfMainMemoryPageProgramThroughBuffer  (
  IN SCARDHANDLE ulHandleCard,
  IN BYTE * pbWriteBuffer,
  IN ULONG ulPageAdr,
  IN ULONG ulByteAdr,
  IN ULONG ulBytesToWrite,
  IN UCHAR ucBufferNumber );
```

Parameters:

`ulHandleCard`	Handle to the DataFlash card.
`pbWriteBuffer`	Pointer to the buffer holding data to be written.
`ulPageAdr`	Page number of the start address.
`ulByteAdr`	Page relative byte number of the start address.
`ulBytesToWrite`	Number of bytes to write.
`ucBufferNumber`	Number 1 or 2 of the buffer to be programmed into the main memory page.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized or DataFlash card type is not supported.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.9.9 SCardDfMainMemoryPageRead

This function allows the user to read data directly from one of the pages in the main memory, bypassing both of the data buffers and leaving the contents of the buffers unchanged. No page boundary is crossed. When the end of a page in the main memory is reached, the read will continue at the beginning of the same page. The application must provide a read buffer with the necessary length.

```
OKERR ENTRY SCardDfMainMemoryPageRead  (
  IN SCARDHANDLE ulHandleCard,
  IN BYTE * pbReadBuffer,
  IN ULONG ulReadBufferSize,
  IN ULONG ulPageAdr,
  IN ULONG ulByteAdr,
  IN ULONG ulBytesToRead,
  OUT PULONG pulBytesReturned );
```

Parameters:

`ulHandleCard`	Handle to the DataFlash card.
`pbReadBuffer`	Pointer to the buffer where the data read will be stored.
`ulReadBufferSize`	Read buffer size.
`ulPageAdr`	Page number of the start address.
`ulByteAdr`	Page relative byte number of the start address.
`ulBytesToRead`	Number of bytes to read.
`pulBytesReturned`	Number of bytes actually read.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized or DataFlash card type is not supported.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.9.10 SCardDfMainMemoryPageToBufferCompare

A page of data in main memory can be compared to the data in buffer 1 or buffer 2. The status register of the DataFlash card is updated. For AT45DB041B this is bit 6. For the other DataFlash types, please refer to the technical specification.

```
OKERR ENTRY SCardDfMainMemoryPageToBufferCompare  (
  IN SCARDHANDLE ulHandleCard,
  IN UCHAR ucBufferNumber,
  IN ULONG ulPageAdr );
```

Parameters:

`ulHandleCard`	Handle to the DataFlash card.
`ucBufferNumber`	Number 1 or 2 of the buffer to be compared.
`ulPageAdr`	Number of the page in the main memory to be compared.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized or DataFlash card type is not supported.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.9.11 SCardDfMainMemoryPageToBufferTransfer

A page of data can be transferred from the main memory page to either buffer 1 or buffer 2. The old content of the selected buffer will be lost.

```
OKERR ENTRY SCardDfMainMemoryPageToBufferTransfer  (
  IN SCARDHANDLE ulHandleCard,
  IN UCHAR ucBufferNumber,
  IN ULONG ulPageAdr );
```

Parameters:

`ulHandleCard`	Handle to the DataFlash card.
`ucBufferNumber`	Number 1 or 2 of the buffer to receive the main memory page.
`ulPageAdr`	Number of the page in the main memory to be transferred.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized or DataFlash card type is not supported.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.9.12 SCardDfPageErase

Erases the specified page in the main memory.

```
OKERR ENTRY SCardDfPageErase  (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulPageAdr );
```

Parameters:

`ulHandleCard`	Handle to the DataFlash card.
`ulPageAdr`	Page number to be erased.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized or DataFlash card type is not supported.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.9.13 SCardDfStatusRegisterRead

The status register can be used to determine the device's ready/busy status, the result of a “main memory page to buffer compare” operation, or the device density. Please refer to the technical specification of the DataFlash card to obtain the meaning of the different bits of the status byte.

```
OKERR ENTRY SCardDfStatusRegisterRead  (
  IN SCARDHANDLE ulHandleCard,
  IN BYTE * pbReadBuffer );
```

Parameters:

`ulHandleCard`	Handle to the DataFlash card.
`pbReadBuffer`	Pointer to the buffer where the status byte will be stored.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized or DataFlash card type is not supported.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.10 Using Contactless memory cards

SCardCL has been planned to support all contactless memory cards. Currently, only the following rf interface cards are supported:

- HID iCLASS cards

The following function is available in the SCardCL module included in this library:

- SCardCLICCTransmit

2.10.1 SCardCLICCTransmit

This function performs the communication between the reader and an iCLASS card. This function is the only gateway for working with an iCLASS card. The parameters must be in accordance with the APDU described in *OMNIKEY Contactless Smart Card Readers Developer Guide* (5321-903), section 5.

```
OKERR ENTRY SCardCLICCTransmit  (
  IN SCARDHANDLE ulHandleCard,
  IN PUCHAR pucSendData,
  IN ULONG ulSendDataBufLen,
  IN OUT PUCHAR pucReceivedData,
  IN OUT PULONG pulReceivedDataBufLen );
```

Parameters:

`ulHandleCard`	Handle to the iCLASS card, provided from the smart card resource manager after connecting the card (`SCardConnect`)
`pucSendData`	The APDU must be according to the specification. Refer to *OMNIKEY Contactless Smart Card Readers Developer Guide* (5321-903).
`ulSendDataBufLen`	Length of the APDU.
`pucReceivedData`	A buffer provided to receive the response.
`pulReceivedDataBufLen`	Length of the provided buffer is given, it returns the actual received bytes.

Returns:

OK Standard Error Codes; see header file **ok.h**.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.11 Using Secure Memory cards with authentication

For more information about the functions that support smart card services, see MSDN.

2.11.1 Open the connection to the card

First, you must call the function `SCardEstablishContext` to establish the resource manager context:

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

If the function returns `SCARD_S_SUCCESS`, the context has been successfully created. You can now list all installed readers in your computer by using `SCardListReaders`:

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

After you have listed the readers you must select one of them. You can do that through a dialog, or in your application. You can then connect to the card using `SCardConnect`:

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

According to the library implementation, the parameter `dwPreferredProtocols` must be `SCARD_PROTOCOL_T0`. After this function has completed successfully, you are connected to the card. You must now initialize it to perform other functions.

2.11.2 Initialize the card

This function initializes library internal data structures according to the Secure Memory `Type` input parameter. Please check if the card type you specify in the call is among the supported types (AT88SC153, AT88SC1608).

```
OKERR SCardSmInit  (
  IN SCARDHANDLE ulHandleCard,
  IN SCARD_SECURE_MEMORY_TYPE Type );
```

`ulHandleCard` must be the card handle you obtained through the pointer `phCard` after calling `SCardConnect`. This handle identifies the connection to the Secure Memory card in the designated reader. After successful completion of `SCardSmInit` you can use the rest of the **scardsyn.dll** library functions.

2.11.3 Close the connection to the card

When you have finished working with the card, you should close the connection using `SCardDisconnect`.

```
LONG SCardDisconnect  (
  IN SCARDHANDLE ulHandleCard,
  IN DWORD dwDisposition );
```

2.12 SCardSM overview

Currently the **scardsyn.dll** library supports only the following Secure Memory with Authentication from ATMEL:

- AT88SC153
- AT88SC1608
- AT88SC101
- AT88SC102
- AT88SC1003

For a full understanding of the operation and functions of these cards, please refer to the corresponding Secure Memory with Authentication AT88SC153, AT88SC1608, AT88SC101, AT88SC102 and AT88SC1003 data sheets.

The following functions are available in the SCardSm module included in this library:

All cards	AT88SC153, AT88SC1608	AT88SC101, AT88SC102 and AT88SC1003
SCardSmInit	SCardSmReadUserZone	SCardSmAT88SC10xRead
	SCardSmWriteUserZone	SCardSmAT88SC10xErase
	SCardSmReadConfigurationZone	SCardSmAT88SC10xEraseAZ
	SCardSmWriteConfigurationZone	SCardSmAT88SC10xCompareSC
	SCardSmReadFuses	SCardSmAT88SC10xBlowFuse
	SCardSmUseSlaveAddress	SCardSmAT88SC10xWrite
	SCardSmVerifyPassword	SCardSmAT88SC10xSetFusPin
	SCardSmWriteFuses	
	SCardSmVerifyAuthentication	
	SCardSmInitializeAuthentication	
	SCardSmSetUserZoneAddress	

2.12.1 Secure Memory Card Types

These constants are used as a Secure Memory Card `Type` input parameter in the call of `SCardSmInit` function:

- AT88SC153
- AT88SC1608
- AT88SC101
- AT88SC102
- AT88SC1003

2.12.2 SCardSmAT88SC10xBlowFuse

(AT88SC10x only) This function blows the Manufacturer, Issuer or ECEN fuse. Blowing fuses is only possible in Security Level 1, so you must set the FUS pin to HIGH in advance by using the function `SCardSmAT88SC10xSetFusPin`.

```
OKERR SCardSmAT88SC10xBlowFuse  (
  IN SCARDHANDLE ulHandleCard,
  IN UCHAR ucFuse );
```

Parameters:

`ulHandleCard`	Handle to the Secure Memory card.
`ucFuse`	Fuse to blow: AT88SC10X_FUSE_MANUFACTURER AT88SC10X_FUSE_ISSUER AT88SC10X_FUSE_ECEN

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized.

`OKERR_NOT_SUPPORTED`: The command is not supported for this card.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.12.3 SCardSmAT88SC10xCompareSC

(AT88SC10x only) This function handles the authentication to the card by comparing the Security Code (two bytes). The attempt counter is returned in `pucAC`. If it is 0xFF, the password verification was successful.

```
OKERR SCardSmAT88SC10xCompareSC  (
  IN SCARDHANDLE ulHandleCard,
  IN PBYTE pbCompareBuffer,
  OUT PUCHAR pucAC );
```

Parameters:

`ulHandleCard`	Handle to the Secure Memory card.
`pbCompareBuffer`	Buffer containing 16 Bit Password (SC).
`pucAC`	Attempt counter (optional: may be NULL).

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized.

`OKERR_PW_WRONG`: Password does not match.

`OKERR_PW_LOCKED`: No password attempts left.

`OKERR_NOT_SUPPORTED`: The command is not supported for this card.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.12.4 SCardSmAT88SC10xErase

(AT88SC10x only) This function erases one or more 16-bit words in memory. In Security Level 2, the application zones can only be erased with the command `SCardSmAT88SC10xEraseAZ`.

```
OKERR SCardSmAT88SC10xErase  (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulWordAddress,
  IN ULONG ulWordsToErase );
```

Parameters:

`ulHandleCard`	Handle to the Secure Memory card.
`ulWordAddress`	Start word to begin erasing.
`ulWordsToErase`	Number of words to erase.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized.

`OKERR_NOT_SUPPORTED`: The command is not supported for this card.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.12.5 SCardSmAT88SC10xEraseAZ

(AT88SC10x only) This function erases a complete application zone. This function only works in Security Level 2. If `fUseEraseCounter==TRUE`, the erase counter is decreased.

```
OKERR SCardSmAT88SC10xEraseAZ  (
  IN SCARDHANDLE ulHandleCard,
  IN UCHAR ucAzNumber,
  IN PBYTE pbCompareBuffer,
  IN BOOL fUseEraseCounter,
  OUT PUCHAR pucEC );
```

Parameters:

`ulHandleCard`	Handle to the Secure Memory card.
`ucAzNumber`	Number of the application zone (1, 2, or 3).
`pbCompareBuffer`	Buffer containing 32/48 Bit Password (EZ1/EZ2/EZ3).
`fUseEraseCounter`	Use Erase Counter (TRUE/FALSE).
`pucEC`	In Security Level 2 the EC2 is returned as a number between 0 and 128 (optional: may be NULL).

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized.

`OKERR_NOT_SUPPORTED`: The command is not supported for this card.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.12.6 SCardSmAT88SC10xRead

(AT88SC10x only) This function reads data bit by bit from the card. It always reads a multiple of 8 bits to fill complete bytes. For memory location, area sizes and access rights, refer to the AT88SC10x data sheets. Some memory areas need authentication to be read. When you try to read such an area without authenticating to the card, 0xFF will be returned in the buffer.

```
OKERR SCardSmAT88SC10xRead  (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulBitAddress,
  IN ULONG ulBitsToRead,
  OUT LPBYTE pbReadBuffer,
  IN ULONG ulReadBufferSize,
  OUT PULONG pulBitsRead );
```

Parameters:

`ulHandleCard`	Handle to the Secure Memory card.
`ulBitAddress`	Start bit to read from.
`ulBitsToRead`	Number of bits to read from the card.
`pbReadBuffer`	Pointer to the read buffer.
`ulReadBufferSize`	Size of the read buffer.
`pulBitsRead`	Number of bits read (optional: may be NULL).

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized.

`OKERR_NOT_SUPPORTED`: The command is not supported for this card.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.12.7 SCardSmAT88SC10xSetFusPin

(AT88SC10x only) Sets C4 (FUS) high (`ucFUS=1`) or low (`ucFUS=0`). The card is in security level 2 if the FUS Pin is low or the Issuer Fuse is blown (0). The card is in security level 1 if the FUS Pin is high and the Issuer Fuse is unblown (1).

```
OKERR SCardSmAT88SC10xSetFusPin  (
  IN SCARDHANDLE ulHandleCard,
  IN UCHAR ucFUS );
```

Parameters:

`ulHandleCard`	Handle to the Secure Memory card.
`ucFUS`	Desired status of the FUS pin (C4).

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized.

`OKERR_NOT_SUPPORTED`: The command is not supported for this card.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.12.8 SCardSmAT88SC10xWrite

(AT88SC10x only) This function allows you to write data from `pbWriteBuffer` bit by bit to the card. Writing starts at `ulBitAddress` and `ulBitsToWrite` are written. Only zeros are written, so if the memory is not erased you must erase it in advance.

```
OKERR SCardSmAT88SC10xWrite  (
  IN SCARDHANDLE ulHandleCard,
  IN ULONG ulBitAddress,
  IN ULONG ulBitsToWrite,
  IN LPBYTE pbWriteBuffer );
```

Parameters:

`ulHandleCard`	Handle to the Secure Memory card.
`ulBitAddress`	Start bit to write.
`ulBitsToWrite`	Number of bits to write.
`pbWriteBuffer`	Pointer to the write buffer.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized.

`OKERR_NOT_SUPPORTED`: The command is not supported for this card.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.12.9 SCardSmInit

This function initializes the shared library for Secure Memory card specified with `Type`. The function must be called once at the beginning, before using any other command, and when a new card is inserted in the reader.

```
OKERR SCardSmInit  (
  IN SCARDHANDLE ulHandleCard,
  IN SCARD_SECURE_MEMORY_TYPE Type );
```

AT88SC153 only: The I2C bus slave address in the most significant 4 bits of the command byte will be set to 0xB0. Any second chip select (i2c-bus device slave) address set so far will be disabled.

AT88SC1608 only: Initializes the user zone address register with 0 by calling `SCardSmSetUserZoneAddress`.

Parameters:

`ulHandleCard`	Handle to the Secure Memory card.
`Type`	Type of the Secure Memory card for which the library will be initialized.

Returns:

`NO_ERROR`: The shared library is initialized.

`OKERR_INIT`: The shared library is not initialized because the Secure Memory card type is not supported.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.12.10 SCardSmInitializeAuthentication

Not supported.

```
OKERR SCardSmInitializeAuthentication  (
  IN SCARDHANDLE ulHandleCard,
  IN PBYTE pbHostRandomNumber );
```

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.12.11 SCardSmReadConfigurationZone

(AT88SC153 and AT88SC1608 only) Reads a number of bytes from the configuration zone. When a secure code password is not successfully verified, some configuration sub-zones will not be able to be read, and instead the internal value 0xFF or the fuse byte will be received from the card. For more details, see the data sheets of the corresponding Secure Memory card.

```
OKERR SCardSmReadConfigurationZone  (
  IN SCARDHANDLE ulHandleCard,
  IN PBYTE pbReadBuffer,
  IN ULONG ulReadBufferSize,
  IN ULONG ulByteAdr,
  IN ULONG ulBytesToRead,
  OUT PULONG pulBytesRead );
```

Parameters:

`ulHandleCard`	Handle to the Secure Memory card.
`pbReadBuffer`	Pointer to the buffer where the data read will be stored.
`ulReadBufferSize`	Read buffer size.
`ulByteAdr`	Address in the configuration zone to start reading from.
`ulBytesToRead`	Number of bytes to read.
`pulBytesRead`	Number of bytes actually read.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized.

`OKERR_NOT_ENOUGH_MEMORY`: Read buffer size provided is smaller than bytes to read.

`OKERR_ILLEGAL_ARGUMENT`: Roll-over to the first byte in the zone is imminent.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.12.12 SCardSmReadFuses

(AT88SC153 and AT88SC1608 only) Reads the fuses byte.

```
OKERR SCardSmReadFuses  (
  IN SCARDHANDLE ulHandleCard,
  IN PBYTE pbReadBuffer );
```

Parameters:

`ulHandleCard`	Handle to the Secure Memory card.
`pbReadBuffer`	Pointer to the buffer where the data read will be stored.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.12.13 SCardSmReadUserZone

(AT88SC153 and AT88SC1608 only) This function reads a number of bytes from the specified user zone. When the “Read Password Enable” bit is set in the access register AR corresponding to the selected user zone, “Verify Password” must be executed with read or write password to allow read operations in the user zone. Password select bits PWx define which of the password sets must be presented to allow access to the user zone.

```
OKERR SCardSmReadUserZone  (
  IN SCARDHANDLE ulHandleCard,
  IN PBYTE pbReadBuffer,
  IN ULONG ulReadBufferSize,
  IN ULONG ulUserZone,
  IN ULONG ulByteAdr,
  IN ULONG ulBytesToRead,
  OUT PULONG pulBytesRead );
```

On AT88SC1608, also executes the command “Set user zone address” (0xB2), in case the user zone parameter is different from the current user zone.

Note: After `SCardSmInit()` function, the current user zone is 0.

Parameters:

`ulHandleCard`	Handle to the Secure Memory card.
`pbReadBuffer`	Pointer to the buffer where the read data is stored.
`ulReadBufferSize`	Read buffer size.
`ulUserZone`	User zone number to read from.
`ulByteAdr`	Address to start reading from, relative to zone begin.
`ulBytesToRead`	Number of bytes to read.
`pulBytesRead`	Number of bytes actually read.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized.

`OKERR_ILLEGAL_ARGUMENT`: Roll-over to the first byte in the page is imminent.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.12.14 SCardSmSetUserZoneAddress

(AT88SC1608 only) This function sets the user zone address register with the address of the user zone which is intended to be referenced with the following user zone read and write commands. The command is for Secure Memory cards from type AT88SC1608 only.

```
OKERR SCardSmSetUserZoneAddress  (
  IN SCARDHANDLE ulHandleCard,
  IN UCHAR ucUserZone );
```

Parameters:

`ulHandleCard`	Handle to the Secure Memory card.
`ucUserZone`	User zone address to be set in the register.

Returns:

OK Standard Error Codes; see header file **ok.h**.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.12.15 SCardSmUseSlaveAddress

(AT88SC153 only) Initializes the shared library to use the specified second slave address to refer to a particular AT88SC153 card.

```
OKERR SCardSmUseSlaveAddress  (
  IN UCHAR ucSlaveAddress );
```

After successful execution of this function, all Secure Memory card commands will have codes with this slave address in the MSb [b7...b4]. Any other cards connected to the same I2C bus, and still having default slave address 0xB0, will not be able to respond to the Secure Memory card commands.

Before using this command, the DEVICE CONFIGURATION REGISTER (at address 0x18 from the configuration space) must be initialized with chip select bits CS3, SC2, SC1, SC0 equal to the most significant bits of the slave address. Address 0xB0 is default primary and doesn't have to be set in the DEVICE CONFIGURATION REGISTER.

If the DEVICE CONFIGURATION REGISTER is not properly initialized before calling this function, and subsequent commands fail, call this function again with the default slave address 0xB0, or call `SCardSmInit` again. The `SCardSmInit` function sets 0xB0 as a slave address in the Secure Memory commands.

This function applies to AT88SC153 Secure Memory cards only, is library internal and performs no physical card operation.

Parameters:

`ucSlaveAddress`	In slave address that will be used with the command codes.

Returns:

`OKERR_NOT_SUPPORTED`: The command is not relevant for the currently initialized Secure Memory card type.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.12.16 SCardSmVerifyAuthentication

Not supported.

```
OKERR SCardSmVerifyAuthentication  (
  IN SCARDHANDLE ulHandleCard,
  IN PBYTE pbHostRandomNumber,
  OUT PUCHAR pucAttemptsCounter );
```

Parameters:

`ulHandleCard`	Handle to the Secure Memory card.
`pucAttemptsCounter`	Authentication attempt counter after the operation.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.12.17 SCardSmVerifyPassword

(AT88SC153 and AT88SC1608 only) This function verifies the password present in the buffer with the password index `ucPasswordIndex` in the password configuration sub-zone. If the password verification is successful, after the operation the corresponding password attempts counter will be reset and will have the value 0xFF. Otherwise, the next higher significant bit from the PAC will be cleared.

```
OKERR SCardSmVerifyPassword  (
  IN SCARDHANDLE ulHandleCard,
  IN PBYTE pbPasswordBuffer,
  IN UCHAR ucPasswordIndex,
  OUT PUCHAR pucAttemptCounter );
```

Parameters:

`ulHandleCard`	Handle to the Secure Memory card.
`pbPasswordBuffer`	Pointer to the buffer holding the 3 byte long password. The bytes are send to the card beginning with the first byte, index 0 of the buffer.
`ucPasswordIndex`	Index of the password set used in the verification. Bit 3 is 0 for write and 1 for read password sets.
`pucAttemptCounter`	Pointer to an address where the new value of the corresponding password attempts counter (PAC) received from the card will be written.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.12.18 SCardSmWriteConfigurationZone

(AT88SC153 and AT88SC1608 only) Writes bytes in the defined configuration zone address. Please refer to the AT88SC153 and AT88SC1608 data sheets to get the address, the meanings and the access rights to particular configuration sub-zones.

The total of write address plus bytes to write can be longer than a write page size, and the write operation can cross a page border. In this case, the write operation is split into pages and several internal write page commands are issued. This process is transparent to the user application. At the end of each internal write page command, a 20 ms delay for the “write cycle time, Twr” is imposed according to the requirements of the AT88SC153 and AT88SC1608 devices.

If the total of write address plus bytes to write is longer than the zone size, an error will be returned indicating possible roll-over to the beginning of the selected user zone. Note that the ability to write on particular addresses from the configuration zone is dependent on successful secure code verification.

This command can also be used for setting a new read and write password for the different user zones.

```
OKERR SCardSmWriteConfigurationZone  (
  IN SCARDHANDLE ulHandleCard,
  IN PBYTE pbWriteBuffer,
  IN ULONG ulByteAdr,
  IN ULONG ulBytesToWrite,
  OUT PULONG pulBytesWritten );
```

Parameters:

`ulHandleCard`	Handle to the Secure Memory card.
`pbWriteBuffer`	Pointer to the buffer where the data read will be stored.
`ulByteAdr`	Address in the configuration zone to start writing.
`ulBytesToWrite`	Number of bytes to write.
`pulBytesWritten`	Number of bytes actually written after execution of the command.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized.

`OKERR_ILLEGAL_ARGUMENT`: Roll-over is imminent.

`OKERR_WRITE_FIO`: The number of bytes actually written is not equal to the intended number of bytes to be written.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.12.19 SCardSmWriteFuses

(AT88SC153 and AT88SC1608 only) Blows a fuse bit from the fuses register in the configuration zone. The fuses are blown (set to 0) sequentially: FAB, CMA, PER. The write fuses command is only enabled when the secure code is verified successfully.

```
OKERR SCardSmWriteFuses  (
  IN SCARDHANDLE ulHandleCard,
  IN PBYTE pucFuseIndex );
```

AT88SC1608: The command has the same code as write configuration zone, but consists only of command and address=0x80 bytes, without a data byte. Each time the command is called, the next available fuse is blown.

Parameters:

`ulHandleCard`	Handle to the Secure Memory card.
`pucFuseIndex`	In AT88SC153 only.

Returns:

OK Standard Error Codes; see header file **ok.h**.

`OKERR_INIT`: The shared library is not initialized.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

2.12.20 SCardSmWriteUserZone

(AT88SC153 and AT88SC1608 only) Writes bytes in the defined user zone.

The total of write address plus bytes to write may be longer than a write page size, and the write operation can cross a page border. In this case the write operation is split into pages and several internal write page commands are issued. This is transparent to the user application. At the end of each internal write page command, a 20 ms delay for the “write cycle time, Twr” is imposed according to the requirements of the AT88SC153 and AT88SC1608 devices.

If the total of write address plus bytes to write is longer than the zone size, an error will be returned indicating possible roll-over to the beginning of the user zone. Note that the ability to write in particular addresses from the user zone is dependent on successful write password verification and settings in the corresponding access register.

```
OKERR SCardSmWriteUserZone  (
  IN SCARDHANDLE ulHandleCard,
  IN PBYTE pbWriteBuffer,
  IN ULONG ulUserZone,
  IN ULONG ulByteAdr,
  IN ULONG ulBytesToWrite,
  OUT PULONG pulBytesWritten );
```

On AT88SC1608, this also executes internally the command 0xB2 “Set user zone address”, in case the user zone parameter is different from the current user zone and the last is updated.

Note: During the personalization (PER=1), the WPE bit in the access register is forced active, and writing in the user zone is only successful after password verification.

Parameters:

`ulHandleCard`	Handle to the Secure Memory card.
`pbWriteBuffer`	Pointer to the buffer where the data read will be stored.
`ulUserZone`	User zone to read from.
`ulByteAdr`	Page relative byte number of the start address.
`ulBytesToWrite`	Number of bytes to write.
`pulBytesWritten`	Number of bytes actually written.

Returns:

OK Standard Error Codes; see header file **ok.h**.

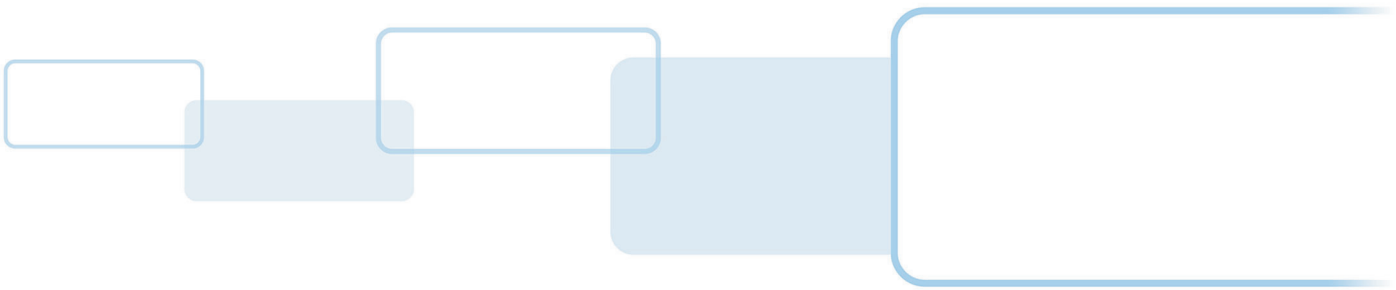
`OKERR_INIT`: The shared library is not initialized.

`OKERR_ILLEGAL_ARGUMENT`: Roll-over is imminent, or an invalid `ulUserZone` number was supplied.

Warning:

This function is supported by 8051 controller based readers only. See *Section 1.5 Cards supported by the OMNIKEY reader*.

This page is intentionally left blank.



hidglobal.com