

MUSCLE PC/SC Lite API

Toolkit API Reference Documentation

Written by David Corcoran corcoran@linuxnet.com

This toolkit and documentation is provided on an as is basis. The author shall not be held responsible for any mishaps caused by the use of this software. For more information please visit <http://www.linuxnet.com>

Contents

1. Introduction	3
2. Overview	3
3. Variable types and definitions	4
4. MUSCLE PC/SC API Routines	5
4.1 SCardEstablishContext: Establish context to PC/SC Resource Manager	6
4.2 SCardReleaseContext: Release context to PC/SC Resource Manager	7
4.3 SCardListReaders: Lists the currently available readers	8
4.4 SCardConnect: Connects and initializes the specified reader	9
4.5 SCardDisconnect: Disconnects from the current reader	10
4.6 SCardBeginTransaction: Begins exclusive access mode for transaction	11
4.7 SCardEndTransaction: Ends exclusive mode for transaction	12
4.8 SCardTransmit: Sends and APDU to the smartcard and gets response	13
4.9 SCardStatus: Returns the current state of the reader/card	15
4.10 SCardGetStatusChange: Blocks for status change	17
4.11 SCardCancel: Releases block in previous function	19
4.12 SCardReadMemory: Reads a buffer from a memory card	20
4.13 SCardWriteMemory: Writes a buffer to a memory card	21
4.14 SCardSetTimeout: Sets the timeout for RPC response	22

MUSCLE PC/SC Toolkit API Reference Documentation

Introduction / Overview

This document contains the reference API calls for communicating to the MUSCLE PC/SC Smartcard Resource Manager. PC/SC is a standard proposed by the PC/SC workgroup which is a conglomerate of representative from major smartcard manufacturers and other companies. This specification tries to abstract the smartcard layer into a high level API so that smartcards and their readers can be accessed in a homogeneous fashion.

This toolkit was written in ANSI C which can be used with most compilers and does NOT use complex and large data structures such as vectors/etc.. The C API emulates the winscard API which is used on the Windows platform. It is contained in the library libpcsc-lite.so which is linked to your application.

I would really like to hear from you. If you have any feedback either on this documentation or on the MUSCLE project please feel free to email me at: corcoran@linuxnet.com

MUSCLE PC/SC Toolkit API Reference Documentation

3. The following is a list of commonly used type definitions in the following API. These definitions and more can be found in the include/pcsclite.h file.

BYTE	unsigned char
USHORT	unsigned short
ULONG	unsigned long
BOOL	short
DWORD	unsigned long
WORD	unsigned long
LONG	long
RESPONSECODE	long
LPCSTR	const char *
SCARDCONTEXT	unsigned long
PSCARDCONTEXT	unsigned long *
LPSCARDCONTEXT	unsigned long *
SCARDHANDLE	unsigned long
PSCARDHANDLE	unsigned long *
LPSCARDHANDLE	unsigned long *
LPCVOID	const void *
LPVOID	void *
LPCBYTE	const unsigned char *
LPBYTE	unsigned char *
LPDWORD	unsigned long *
LPSTR	char *
LPCWSTR	char *

The following is a list of commonly used errors. Since different cards produce different errors they must map over to these error messages.

SCARD_E_UNSUPPORTED_INTERFACE	SCARD_E_UNSUPPORTED_FEATURE
SCARD_E_NOTIMPL	SCARD_E_UNSUPPORTED_FUNCTION
SCARD_E_INSUFFICIENT_BUFFER	SCARD_E_INVALID_ATR
SCARD_E_INVALID_HANDLE	SCARD_E_INVALID_PARAMETER
SCARD_E_INVALID_TARGET	SCARD_E_INVALID_VALUE
SCARD_F_COMM_ERROR	SCARD_F_INTERNAL_ERROR
SCARD_E_UNKNOWN_READER	SCARD_E_TIMEOUT
SCARD_E_SHARING_VIOLATION	SCARD_E_NO_SMARTCARD
SCARD_E_UNKNOWN_CARD	SCARD_E_NOT_READY
SCARD_E_SYSTEM_CANCELLED	SCARD_E_NOT_TRANSACTED
SCARD_E_READER_UNAVAILABLE	SCARD_F_UNKNOWN_ERROR
SCARD_W_UNSUPPORTED_CARD	SCARD_W_UNRESPONSIVE_CARD
SCARD_W_UNPOWERED_CARD	SCARD_W_RESET_CARD
SCARD_W_REMOVED_CARD	SCARD_W_INSERTED_CARD
SCARD_E_UNKNOWN_READER	SCARD_E_TIMEOUT
SCARD_E_NO_SMARTCARD	SCARD_E_UNKNOWN_CARD
SCARD_E_PROTO_MISMATCH	SCARD_E_SYSTEM_CANCELLED
SCARD_E_PCI_TOO_SMALL	SCARD_E_READER_UNSUPPORTED
SCARD_E_DUPLICATE_READER	SCARD_E_CARD_UNSUPPORTED
SCARD_E_NO_SERVICE	SCARD_E_SERVICE_STOPPED

Section 4

MUSCLE PC/SC API Routines

These routines specified here are winscard routines like those in the winscard API provided under Windows ® . These are compatible with the Microsoft ® API calls. This list of calls is mainly an abstraction of readers. This API has some extended functions for manipulating memory cards. This will be denoted in the documentation. It gives a common API for communication to most readers in a homogeneous fashion. Since all functions can produce a wide array of errors, please refer to page 4 for a list of error returns.

MUSCLE PC/SC Toolkit API Reference Documentation

Synopsis:

```
#include <winscard.h>
```

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

Parameters:

dwScope:	IN	Scope of the establishment. This can either be a local or remote connection
pvReserved1:	IN	Reserved for future use. Can be used for remote connection.
pvReserved2:	IN	Reserved for future use.
phContext:	OUT	Returned reference to this connection.

Description:

This function creates a communication context to the PC/SC Resource Manager. This must be the first function called in a PC/SC application.

Value of dwScope	Meaning
SCARD_SCOPE_USER	Not used.
SCARD_SCOPE_TERMINAL	Not used.
SCARD_SCOPE_SYSTEM	Services on the local machine.
SCARD_SCOPE_GLOBAL	Services are on a remote machine.

Note: If SCARD_SCOPE_GLOBAL is used then pvReserved1 is a string which is the hostname of the machine which the Resource Manager services reside. If NULL is specified then it defaults to the localhost.

Example:

```
SCARDCONTEXT hContext;
LONG rv;

rv = SCardEstablishContext( SCARD_SCOPE_SYSTEM, NULL, NULL, &hContext );
```

Returns:

SCARD_S_SUCCESS	- Successful
SCARD_E_INVALID_VALUE	- Invalid scope type passed.

MUSCLE PC/SC Toolkit API Reference Documentation

Synopsis:

```
#include <winscard.h>
```

```
LONG SCardReleaseContext( SCARDCONTEXT hContext );
```

Parameters:

hContext: IN Connection context to be closed.

Description:

This function destroys a communication context to the PC/SC Resource Manager. This must be the last function called in a PC/SC application.

Example:

```
SCARDCONTEXT hContext;  
LONG rv;  
  
rv = SCardEstablishContext( SCARD_SCOPE_SYSTEM, NULL, NULL, &hContext );  
rv = SCardReleaseContext( hContext );
```

Returns:

SCARD_S_SUCCESS	- Successful
SCARD_E_INVALID_HANDLE	- Invalid hContext handle.

MUSCLE PC/SC Toolkit API Reference Documentation

Synopsis:

```
#include <winscard.h>
```

```
LONG SCardListReaders( SCARDCONTEXT hContext, LPCSTR mszGroups,
                      LPSTR mszReaders, LPDWORD pcchReaders );
```

Parameters:

hContext:	IN	Connection context to the PC/SC Resource Manager.
mszGroups	IN	List of groups to list readers (not used)
mszReaders	OUT	Multi-string with list of readers.
pcchReaders	INOUT	Size of multi-string buffer including NULL's.

Description:

This function returns a list of currently available readers on the system. mszReaders is a pointer to a character string which will be allocated by the application. If the application sends mszGroups and mszReaders as NULL then this function will return the size of the buffer needed to allocate in pcchReaders. The reader names will be a multi-string and separated by a NULL character and ended by a double NULL. "ReaderA\0ReaderB\0\0"

Example:

```
SCARDCONTEXT hContext;
LPSTR mszGroups;
LPSTR mszReaders;
DWORD dwReaders;
LONG rv;

rv = SCardEstablishContext( SCARD_SCOPE_SYSTEM, NULL, NULL, &hContext );
rv = SCardListReaders( hContext, NULL, NULL, &dwReaders );
mszReaders = (LPSTR)malloc(sizeof(char)*dwReaders);
rv = SCardListReaders( hContext, mszGroups, &mszReaders, &dwReaders );
```

Returns:

SCARD_S_SUCCESS	- Successful
SCARD_E_INVALID_HANDLE	- Invalid Scope Handle
SCARD_E_INSUFFICIENT_BUFFER	- Reader buffer not large enough

MUSCLE PC/SC Toolkit API Reference Documentation

Synopsis:

```
#include <winscard.h>
```

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

Parameters:

hContext:	IN	Connection context to the PC/SC Resource Manager.
szReader:	IN	Reader name to connect to.
dwShareMode:	IN	Mode of connection type: exclusive or shared.
dwPreferredProtocols	IN	Desired protocol use.
phCard	OUT	Handle to this connection.
pdwActiveProtocol	OUT	Established protocol to this connection.

Description:

This function establishes a connection to the friendly name of the reader specified in szReader. The first connection will power up and perform a reset on the card.

Value of dwShareMode

SCARD_SHARE_SHARED
SCARD_SHARE_EXCLUSIVE

Meaning

This application will allow others to share the reader.
This application will NOT allow others to share the reader.

Value of dwPreferredProtocols

SCARD_PROTOCOL_T0
SCARD_PROTOCOL_T1
SCARD_PROTOCOL_RAW

Meaning

Use the T=0 protocol.
Use the T=1 protocol
Use with memory type cards.

Example:

```
SCARDCONTEXT hContext;
SCARDHANDLE hCard
DWORD dwActiveProtocol;
LONG rv;

rv = SCardEstablishContext( SCARD_SCOPE_SYSTEM, NULL, NULL, &hContext );
rv = SCardConnect( hContext, "Reader X", SCARD_SHARE_SHARED,
                  SCARD_PROTOCOL_T0, &hCard, &dwActiveProtocol );
```

Returns:

SCARD_S_SUCCESS	- Successful
SCARD_E_NOT_READY	- Could not allocate the desired port.
SCARD_E_INVALID_VALUE	- Invalid sharing mode, requested protocol, or reader name.
SCARD_E_READER_UNAVAILABLE	- Could not power up the reader or card.
SCARD_E_UNSUPPORTED_FEATURE	- Protocol not supported.
SCARD_E_SHARING_VIOLATION	- Someone else has exclusive rights.
SCARD_E_INVALID_HANDLE	- Invalid hContext handle.

MUSCLE PC/SC Toolkit API Reference Documentation

Synopsis:

```
#include <winscard.h>
```

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

Parameters:

hCard:	IN	Connection made from SCardConnect.
dwDisposition	IN	Reader function to execute.

Description:

This function terminates a connection to the connection made through SCardConnect. dwDisposition can have the following values:

Value of dwDisposition	Meaning
SCARD_LEAVE_CARD	Do nothing.
SCARD_RESET_CARD	Reset the card.
SCARD_UNPOWER_CARD	Unpower the card.
SCARD_EJECT_CARD	Eject the card.

Example:

```
SCARDCONTEXT hContext;
SCARDHANDLE hCard
DWORD dwActiveProtocol;
LONG rv;

rv = SCardEstablishContext( SCARD_SCOPE_SYSTEM, NULL, NULL, &hContext );
rv = SCardConnect( hContext, "Reader X", SCARD_SHARE_SHARED,
                  SCARD_PROTOCOL_T0, &hCard, &dwActiveProtocol );
rv = SCardDisconnect( hCard, SCARD_UNPOWER_CARD );
```

Returns:

SCARD_S_SUCCESS	- Successful
SCARD_E_INVALID_HANDLE	- Invalid hCard handle.
SCARD_E_INVALID_VALUE	- Invalid dwDisposition.

MUSCLE PC/SC Toolkit API Reference Documentation

Synopsis:

```
#include <winscard.h>
```

```
LONG SCardBeginTransaction( SCARDHANDLE hCard );
```

Parameters:

hCard: IN Connection made from SCardConnect.

Description:

This function establishes a temporary exclusive access mode for doing a series of commands or transaction. You might want to use this when you are selecting a few files and then writing a large file so you can make sure that another application will not change the current file. If another application has a lock on this reader or this application is in SCARD_SHARE_EXCLUSIVE there will be no action taken.

Example:

```
SCARDCONTEXT hContext;
SCARDHANDLE hCard
DWORD dwActiveProtocol;
LONG rv;

rv = SCardEstablishContext( SCARD_SCOPE_SYSTEM, NULL, NULL, &hContext );
rv = SCardConnect( hContext, "Reader X", SCARD_SHARE_SHARED,
                  SCARD_PROTOCOL_T0, &hCard, &dwActiveProtocol );
rv = SCardBeginTransaction( hCard );

/* Do some transmit commands */
```

Returns:

SCARD_S_SUCCESS	- Successful
SCARD_E_INVALID_HANDLE	- Invalid hCard handle.
SCARD_E_SHARING_VIOLATION	- Someone else has exclusive rights.

MUSCLE PC/SC Toolkit API Reference Documentation

Synopsis:

```
#include <winscard.h>
```

```
LONG SCardEndTransaction( SCARDHANDLE hCard, DWORD dwDisposition );
```

Parameters:

hCard:	IN	Connection made from SCardConnect.
dwDisposition	IN	Action to be taken on the reader.

Description:

This function ends a previously begun transaction. The calling application must be the owner of the previously begun transaction or an error will occur. dwDisposition can have the following values: The disposition action is not currently used in this release.

Value of dwDisposition	Meaning
SCARD_LEAVE_CARD	Do nothing.
SCARD_RESET_CARD	Reset the card.
SCARD_UNPOWER_CARD	Unpower the card.
SCARD_EJECT_CARD	Eject the card.

Example:

```
SCARDCONTEXT hContext;
SCARDHANDLE hCard
DWORD dwActiveProtocol;
LONG rv;

rv = SCardEstablishContext( SCARD_SCOPE_SYSTEM, NULL, NULL, &hContext );
rv = SCardConnect( hContext, "Reader X", SCARD_SHARE_SHARED,
                  SCARD_PROTOCOL_T0, &hCard, &dwActiveProtocol );
rv = SCardBeginTransaction( hCard );

/* Do some transmit commands */

rv = SCardEndTransaction( hCard, SCARD_LEAVE_CARD );
```

Returns:

SCARD_S_SUCCESS	- Successful
SCARD_E_SHARING_VIOLATION	- Someone else has exclusive rights.
SCARD_E_INVALID_HANDLE	- Invalid hCard handle.

MUSCLE PC/SC Toolkit API Reference Documentation

Synopsis:

```
#include <winscard.h>
```

```
LONG SCardTransmit( SCARDHANDLE hCard, LPCSCARD_IO_REQUEST pioSendPci,
                    LPCBYTE pbSendBuffer, DWORD cbSendLength,
                    LPCSCARD_IO_REQUEST pioRecvPci,
                    LPBYTE pbRecvBuffer, LPDWORD pcbRecvLength );
```

Parameters:

hCard:	IN	Connection made from SCardConnect.
pioSendPci:	INOUT	Structure of protocol information.
pbSendBuffer:	IN	APDU to send to the card.
cbSendLength:	IN	Length of the APDU.
pioRecvPci:	INOUT	Structure of protocol information.
pbRecvBuffer:	OUT	Response from the card.
pcbRecvLength:	INOUT	Length of the response.

Description:

This function sends an APDU to the smartcard contained in the reader connected to by SCardConnect. The card responds from the APDU and stores this response in pbRecvBuffer and it's length in pcbRecvLength. SendPci and RecvPci are structures containing the following:

```
typedef struct {
    DWORD dwProtocol;      /* SCARD_PROTOCOL_T0 or SCARD_PROTOCOL_T1 */
    DWORD cbPciLength;     /* Length of this structure - not used */
} SCARD_IO_REQUEST;
```

Value of pioSendPci

SCARD_PCI_T0

SCARD_PCI_T1

Meaning

Pre defined T=0 PCI structure

Pre defined T=1 PCI structure

Example:

```

LONG rv;
SCARDCONTEXT hContext;
SCARDHANDLE hCard
DWORD dwActiveProtocol, dwSendLength, pcbRecvLength;
SCARD_IO_REQUEST pioRecvPci;
BYTE pbRecvBuffer[10];
BYTE pbSendBuffer = { 0xC0, 0xA4, 0x00, 0x00, 0x02, 0x3F, 0x00 };

dwSendLength = 7;

rv = SCardEstablishContext( SCARD_SCOPE_SYSTEM, NULL, NULL, &hContext );
rv = SCardConnect( hContext, "Reader X", SCARD_SHARE_SHARED,
                  SCARD_PROTOCOL_T0, &hCard, &dwActiveProtocol );
rv = SCardTransmit( hCard, SCARD_PCI_T0, pbSendBuffer, dwSendLength, &pioRecvPci,
                  pbRecvBuffer, &pcbRecvLength );

```

Returns:

SCARD_S_SUCCESS	- Successful
SCARD_E_NOT_TRANSACTED	- APDU exchange not successful.
SCARD_E_INVALID_HANDLE	- Invalid hCard handle.
SCARD_E_PROTO_MISMATCH	- Connect protocol is different than desired.
SCARD_E_INVALID_VALUE	- Invalid Protocol, reader name, etc.

MUSCLE PC/SC Toolkit API Reference Documentation

Synopsis:

```
#include <winscard.h>
```

```
LONG SCardStatus( SCARDHANDLE hCard, LPSTR szReaderName,
                  LPDWORD pcchReaderLen, LPDWORD pdwState,
                  LPDWORD pdwProtocol, LPBYTE pbAtr,
                  LPDWORD pcbAtrLen );
```

Parameters:

hCard:	IN	Connection made from SCardConnect
szReaderName	INOUT	Friendly name of this reader.
pcchReaderLen	INOUT	Size of the szReaderName multi-string
pdwState	OUT	Current state of this reader
pdwProtocol	OUT	Current protocol of this reader
pbAtr	OUT	Current ATR of a card in this reader
pcbAtrLen	OUT	Length of ATR

Description:

This function returns the current status of the reader connected to by hCard. It's friendly name will be stored in szReaderName. pcchReaderLen will be the size of the allocated buffer for szReaderName. If this is too small the function will return with the necessary size in pcchReaderLen. The current state, and protocol will be stored in pdwState and pdwProtocol respectively. pdwState is a DWORD possibly OR 'd with the following values:

Value of pdwState

SCARD_ABSENT

SCARD_PRESENT

SCARD_SWALLOWED

SCARD_POWERED

SCARD_NEGOTIABLEMODE

SCARD_SPECIFICMODE

Meaning

There is no card in the reader.

There is a card in the reader, but it has not been moved into position for use.

There is a card in the reader in position for use. The card is not powered.

Power is being provided to the card, but the reader driver is unaware of the mode of the card.

The card has been reset and is awaiting PTS negotiation.

The card has been reset and specific communication protocols have been established.

Value of dwPreferredProtocols

SCARD_PROTOCOL_T0

SCARD_PROTOCOL_T1

Meaning

Use the T=0 protocol.

Use the T=1 protocol.

Example:

```

SCARDCONTEXT hContext;
SCARDHANDLE hCard;
DWORD dwActiveProtocol, cReaders;
DWORD dwState, dwProtocol, dwAtrLen;
BYTE pbAtr[MAX_ATR_SIZE]
LPSTR mszReaders;
LONG rv;

rv = SCardEstablishContext( SCARD_SCOPE_SYSTEM, NULL, NULL, &hContext );
rv = SCardConnect( hContext, "Reader X", SCARD_SHARE_SHARED,
                  SCARD_PROTOCOL_T0, &hCard, &dwActiveProtocol );
mszReaders = (LPSTR)malloc(sizeof(char)*50);
rv=SCardStatus( hCard, mszReaders, 50, &dwState, &dwProtocol, pbAtr, &dwAtrLen );

```

Returns:

SCARD_S_SUCCESS	- Successful
SCARD_E_INVALID_HANDLE	- Invalid hCard handle
SCARD_E_INSUFFICIENT_BUFFER	- Not enough allocated memory for szReaderName

MUSCLE PC/SC Toolkit API Reference Documentation

Synopsis:

```
#include <winscard.h>
```

```
LONG SCardGetStatusChange( SCARDCONTEXT hContext, DWORD dwTimeout,
                           LPSCARD_READERSTATE rgReaderStates,
                           DWORD cReaders );
```

Parameters:

hContext:	IN	Connection context to the PC/SC Resource Manager.
dwTimeout	IN	Maximum block waiting time for status change, zero for infinite.
rgReaderStates	INOUT	Structures of readers with current states.
cReaders	IN	Number of structures.

Description:

This function receives a structure or list of structures containing reader names. It then blocks for a change in state to occur on any of the OR 'd values contained in dwCurrentState for a maximum blocking time of dwTimeout or forever. The new event state will be contained in dwEventState. A status change might be a card insertion or removal event, a change in ATR, etc. This function currently only takes 1 reader as an argument.

```
typedef struct {
    LPCTSTR szReader;          /* Reader name */
    LPVOID pvUserData;         /* User defined data */
    DWORD dwCurrentState;      /* Current state of reader */
    DWORD dwEventState;        /* Reader state after a state change */
    DWORD cbAtr;               /* ATR Length, usually MAX_ATR_SIZE */
    BYTE rgbAtr[36];           /* ATR Value */
} SCARD_READERSTATE;

typedef SCARD_READERSTATE *PSCARD_READERSTATE, *LPSCARD_READERSTATE;
```

Value of dwCurrentState/dwEventState

SCARD_STATE_UNAWARE

SCARD_STATE_IGNORE

SCARD_STATE_CHANGED

SCARD_STATE_UNKNOWN

Meaning

The application is unaware of the current state, and would like to know. The use of this value results in an immediate return from state transition monitoring services. This is represented by all bits set to zero.

This reader should be ignored.

There is a difference between the state believed by the application, and the state known by the resource manager. When this bit is set, the application may assume a significant state change has occurred on this reader.

The given reader name is not recognized by the resource manager. If this bit is set, then SCARD_STATE_CHANGED and SCARD_STATE_IGNORE will also be set.

Value of dwCurrentState/dwEventState	Meaning
SCARD_STATE_UNAVAILABLE	The actual state of this reader is not available. If this bit is set, then all the following bits are clear.
SCARD_STATE_EMPTY	There is no card in the reader. If this bit is set, all the following bits will be clear.
SCARD_STATE_PRESENT	There is a card in the reader.
SCARD_STATE_ATRMATCH	There is a card in the reader with an ATR matching one of the target cards. If this bit is set, SCARD_STATE_PRESENT will also be set. This bit is only returned on the SCardLocateCards function.
SCARD_STATE_EXCLUSIVE	The card in the reader is allocated for exclusive use by another application. If this bit is set, SCARD_STATE_PRESENT will also be set.
SCARD_STATE_INUSE	The card in the reader is in use by one or more other applications, but may be connected to in shared mode. If this bit is set, SCARD_STATE_PRESENT will also be set.
SCARD_STATE_MUTE	There is an unresponsive card in the reader.

Example:

```

SCARDCONTEXT hContext;
SCARDHANDLE hCard
DWORD dwActiveProtocol, cReaders;
SCARD_READERSTATE rgReaderStates;
LONG rv;

rv = SCardEstablishContext( SCARD_SCOPE_SYSTEM, NULL, NULL, &hContext );

rgReaderStates.szReader      = strdup("Reader X");
rgReaderStates.dwCurrentState = SCARD_STATE_EMPTY;

rv = SCardGetStatusChange( hContext, 0, rgReaderStates, &cReaders );
rv = SCardConnect( hContext, "Reader X", SCARD_SHARE_SHARED,
                  SCARD_PROTOCOL_T0, &hCard, &dwActiveProtocol );

```

Returns:

SCARD_S_SUCCESS	- Successful
SCARD_E_INVALID_VALUE	- Invalid States, reader name, etc.
SCARD_E_INVALID_HANDLE	- Invalid hContext handle.

MUSCLE PC/SC Toolkit API Reference Documentation

Synopsis:

```
#include <winscard.h>

LONG SCardCancel( SCARDCONTEXT hContext );
```

Parameters:

hContext: IN Connection context to the PC/SC Resource Manager.

Description:

This function cancels all pending blocking requests on the GetStatusChange function.

Example:

```
SCARDCONTEXT hContext;
DWORD cReaders;
SCARD_READERSTATE rgReaderStates;

LONG rv;

rv = SCardEstablishContext( SCARD_SCOPE_SYSTEM, NULL, NULL, &hContext );

rgReaderStates.szReader        = strdup("Reader X");
rgReaderStates.dwCurrentState = SCARD_STATE_EMPTY;

/* Spawn off thread for following function */
rv = SCardGetStatusChange( hContext, 0, rgReaderStates, &cReaders );

rv = SCardCancel( hContext );
```

Returns:

SCARD_S_SUCCESS	- Successful
SCARD_E_INVALID_HANDLE	- Invalid hContext handle.

MUSCLE PC/SC Toolkit API Reference Documentation

Synopsis:

```
#include <winscard.h>
```

```
LONG SCardReadMemory( SCARDHANDLE hCard, DWORD dwCardType,
                      DWORD dwCardSize, DWORD dwAddress,
                      LPBYTE pBuffer, DWORD dwLength );
```

Parameters:

hCard:	IN	Connection context to the PC/SC Resource Manager.
dwCardType:	IN	Type of memory card used.
dwCardSize:	IN	Total size of card. (32kb is 32)
dwAddress:	IN	Starting address to read.
pBuffer:	OUT	Buffer for data.
dwLength:	IN	Length of data to read.

Description:

This function reads data from a memory type smartcard. This function is an extension to the API and is only supported on readers that implement it. The SCardConnect protocol must be set to SCARD_PROTOCOL_RAW.

Value of dwCardType

SCARD_TYPE_I2C
SCARD_TYPE_2WIRE
SCARD_TYPE_3WIRE

Meaning

I2C Memory card being used.
2 Wire Memory card being used.
3 Wire Memory card being used.

Example:

```
SCARDCONTEXT hContext;
SCARDHANDLE hCard
DWORD dwActiveProtocol;
BYTE pBuffer[50];
LONG rv;

rv = SCardEstablishContext( SCARD_SCOPE_SYSTEM, NULL, NULL, &hContext );
rv = SCardConnect( hContext, "Reader X", SCARD_SHARE_SHARED,
                  SCARD_PROTOCOL_RAW, &hCard, &dwActiveProtocol );
rv = SCardReadMemory( hCard, SCARD_TYPE_I2C, 32, 0, pBuffer, 30 );
```

Returns:

SCARD_S_SUCCESS	- Successful
SCARD_E_INVALID_HANDLE	- Invalid hContext handle.
SCARD_E_UNSUPPORTED_FEATURE	- Feature not supported in this reader.

MUSCLE PC/SC Toolkit API Reference Documentation

Synopsis:

```
#include <winscard.h>
```

```
LONG SCardWriteMemory( SCARDHANDLE hCard, DWORD dwCardType,
                      DWORD dwCardSize, DWORD dwAddress,
                      LPCBYTE pcbBuffer, DWORD dwLength );
```

Parameters:

hCard:	IN	Connection context to the PC/SC Resource Manager.
dwCardType:	IN	Type of memory card used.
dwCardSize:	IN	Total size of card. (32kb is 32)
dwAddress:	IN	Starting address to read.
pcbBuffer:	IN	Buffer for data.
dwLength:	IN	Length of data to read.

Description:

This function writes data to a memory type smartcard. This function is an extension to the API and is only supported on readers that implement it. The SCardConnect protocol must be set to SCARD_PROTOCOL_RAW.

Value of dwCardType

SCARD_TYPE_I2C
SCARD_TYPE_2WIRE
SCARD_TYPE_3WIRE

Meaning

I2C Memory card being used.
2 Wire Memory card being used.
3 Wire Memory card being used.

Example:

```
SCARDCONTEXT hContext;
SCARDHANDLE hCard
DWORD dwActiveProtocol;
BYTE pbBuffer[50];
LONG rv;

rv = SCardEstablishContext( SCARD_SCOPE_SYSTEM, NULL, NULL, &hContext );
rv = SCardConnect( hContext, "Reader X", SCARD_SHARE_SHARED,
                  SCARD_PROTOCOL_RAW, &hCard, &dwActiveProtocol );

/* Fill pbBuffer with desired information */
rv = SCardWriteMemory( hCard, SCARD_TYPE_I2C, 32, 0, pbBuffer, 30 );
```

Returns:

SCARD_S_SUCCESS	- Successful
SCARD_E_INVALID_HANDLE	- Invalid hContext handle.
SCARD_E_UNSUPPORTED_FEATURE	- Feature not supported in this reader.

MUSCLE PC/SC Toolkit API Reference Documentation

Synopsis:

```
#include <winscard.h>
```

```
LONG SCardSetTimeout( SCARDCONTEXT hContext, DWORD dwTimeout );
```

Parameters:

hContext:	IN	Connection context to the PC/SC Resource Manager.
dwTimeout	IN	New timeout value.

Description:

This function updates the working waiting time that RPC uses when waiting for a server function to return. This needs to be updated when a card command is sent that might take more time than usual.

Example:

```
SCARDCONTEXT hContext;  
LONG rv;  
  
rv = SCardEstablishContext( SCARD_SCOPE_SYSTEM, NULL, NULL, &hContext );  
rv = SCardSetTimeout( hContext, 50 ); /* 50 second timeout */
```

Returns:

SCARD_S_SUCCESS	- Successful
SCARD_E_INVALID_HANDLE	- Invalid hContext handle.