MUSCLE PC/SC Lite API

Toolkit API Reference Documentation

Written by David Corcoran corcoran@linuxnet.com
Last Modified: March 8, 2001

Contents

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 libpcsclite.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

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_NOTIMPL
SCARD_E_INSUFFICIENT_BUFFER
SCARD_E_INVALID_HANDLE
SCARD_E_INVALID_TARGET
SCARD_F_COMM_ERROR
SCARD_E_UNKNOWN_READER
SCARD_E_SHARING_VIOLATION
SCARD_E_UNKNOWN_CARD
SCARD_E_SYSTEM_CANCELLED
SCARD_E_READER_UNAVAILABLE
SCARD_W_UNSUPPORTED_CARD
SCARD_W_UNPOWERED_CARD
SCARD_W_REMOVED_CARD
SCARD_E_UNKNOWN_READER
SCARD_E_NO_SMARTCARD
SCARD_E_PROTO_MISMATCH
SCARD_E_PCI_TOO_SMALL
SCARD_E_DUPLICATE_READER
SCARD_E_NO_SERVICE
SCARD_E_UNSUPPORTED_FEATURE
SCARD_E_UNSUPPORTED_FUNCTION
SCARD_E_INVALID_ATR
SCARD_E_INVALID_PARAMETER
SCARD_E_INVALID_VALUE
SCARD_F_INTERNAL_ERROR
SCARD_E_TIMEOUT
SCARD_E_NO_SMARTCARD
SCARD_E_NOT_READY
SCARD_E_NOT_TRANSACTED
SCARD_F_UNKNOWN_ERROR
SCARD_W_UNRESPONSIVE_CARD
SCARD_W_RESET_CARD
SCARD_W_INSERTED_CARD
SCARD_E_TIMEOUT
SCARD_E_UNKNOWN_CARD
SCARD_E_SYSTEM_CANCELLED
SCARD_E_READER_UNSUPPORTED
SCARD_E_CARD_UNSUPPORTED
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. 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. For a human readable representation of an error the function pcsc_stringify_error is declared in debuglog.h

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.

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	**Meaning**
SCARD_SHARE_SHARED	This application will allow others to share the reader.
SCARD_SHARE_EXCLUSIVE	This application will NOT allow others to share the reader.

Value of dwPreferredProtocols	**Meaning**
SCARD_PROTOCOL_T0	Use the T=0 protocol.
SCARD_PROTOCOL_T1	Use the T=1 protocol
SCARD_PROTOCOL_RAW	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 SCardReconnect( SCARDHANDLE hCard, DWORD dwShareMode,
                     DWORD dwPreferredProtocols DWORD dwInitialization,
                     LPDWORD pdwActiveProtocol );
```

Parameters:

hCard:	IN	Handle to a previous call to connect.
dwShareMode:	IN	Mode of connection type: exclusive/shared.
dwPreferredProtocols	IN	Desired protocol use.
dwInitialization	IN	Desired action taken on the card/reader.
pdwActiveProtocol	OUT	Established protocol to this connection.

Description:

This function reestablishes a connection to a reader that was previously connected to using SCardConnect. In a multi application environment it is possible for an application to reset the card in shared mode. When this occurs any other application trying to access certain commands will be returned the value SCARD_W_RESET_CARD. When this occurs SCardReconnect must be called in order to acknowledge that the card was reset and allow it to change it's state accordingly.

Value of dwShareMode	**Meaning**
SCARD_SHARE_SHARED	This application will allow others to share the reader.
SCARD_SHARE_EXCLUSIVE	This application will NOT allow others to share the reader.

Value of dwPreferredProtocols	**Meaning**
SCARD_PROTOCOL_T0	Use the T=0 protocol.
SCARD_PROTOCOL_T1	Use the T=1 protocol
SCARD_PROTOCOL_RAW	Use with memory type cards.

Value of dwInitialization	**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;
BYTE pbRecvBuffer[10];
BYTE pbSendBuffer = { 0xC0, 0xA4, 0x00, 0x00, 0x02, 0x3F, 0x00 };
rv = SCardEstablishContext( SCARD_SCOPE_SYSTEM, NULL, NULL, &hContext );
rv = SCardConnect(hContext, “Reader X”, SCARD_SHARE_SHARED,
                SCARD_PROTOCOL_T0, &hCard, &dwActiveProtocol );
dwSendLength = 7;
rv = SCardTransmit( hCard, SCARD_PCI_T0, pbSendBuffer, dwSendLength,
                &pioRecvPci, pbRecvBuffer, &pcbRecvLength );
/* Card has been reset by another application */
if ( rv == SCARD_W_RESET_CARD ) {
rv = SCardReconnect( hCard, SCARD_SHARE_SHARED, SCARD_PROTOCOL_T0,
           SCARD_RESET_CARD, &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	- The reader has been removed.
SCARD_E_UNSUPPORTED_FEATURE	- Protocol not supported.
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 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.
SCARD_E_READER_UNAVAILABLE	- The reader has been removed.

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.

SCARD_E_READER_UNAVAILABLE	- The reader has been removed.

MUSCLE PC/SC Toolkit API Reference Documentation

Synopsis:

#include <winscard.h>

```
LONG SCardTransmit( SCARDHANDLE hCard, LPCSCARD_IO_REQUEST pioSendPci,
                    LPCBYTE pbSendBuffer, DWORD cbSendLength,
                    LPSCARD_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	**Meaning**
SCARD_PCI_T0	Pre defined T=0 PCI structure
SCARD_PCI_T1	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.
SCARD_E_READER_UNAVAILABLE	- The reader has been removed.
SCARD_W_RESET_CARD	- The card has been reset by another application.
SCARD_W_REMOVED_CARD	- The card has been removed from the reader.

MUSCLE PC/SC Toolkit API Reference Documentation

Synopsis:

#include <winscard.h>

```
LONG SCardControl( SCARDHANDLE hCard, LPCBYTE pbSendBuffer,
                   DWORD cbSendLength, LPBYTE pbRecvBuffer,
                   LPDWORD pcbRecvLength );
```

Parameters:

hCard:	IN	Connection made from SCardConnect.
pbSendBuffer:	IN	Command to send to the reader.
cbSendLength:	IN	Length of the command.
pbRecvBuffer:	OUT	Response from the reader
pcbRecvLength:	INOUT	Length of the response.

Description:

This function sends a command directly to the IFD Handler to be processed by the reader. This is useful for creating client side reader drivers for functions like PIN pads, biometrics, or other extensions to the normal smartcard reader that are not normally handled by PC/SC.

Example:

```
LONG rv;
SCARDCONTEXT hContext;
SCARDHANDLE hCard
DWORD dwActiveProtocol, dwSendLength, pcbRecvLength;
BYTE pbRecvBuffer[10];
BYTE pbSendBuffer = { 0x06, 0x00, 0x0A, 0x01, 0x01, 0x10 0x00 };

rv = SCardEstablishContext( SCARD_SCOPE_SYSTEM, NULL, NULL, &hContext );
rv = SCardConnect( hContext, “Reader X”, SCARD_SHARE_SHARED,
                   SCARD_PROTOCOL_RAW &hCard, &dwActiveProtocol );
rv = SCardControl( hCard, pbSendBuffer, 7,
                   pbRecvBuffer, &pcbRecvLength );
```

Returns:

SCARD_S_SUCCESS	- Successful
SCARD_E_NOT_TRANSACTED	- Data exchange not successful.
SCARD_E_INVALID_HANDLE	- Invalid hCard handle.
SCARD_E_INVALID_VALUE	- Invalid value was presented.
SCARD_E_READER_UNAVAILABLE	- The reader has been removed.
SCARD_W_RESET_CARD	- The card has been reset by another application.
SCARD_W_REMOVED_CARD	- The card has been removed from the reader.

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	**Meaning**
SCARD_ABSENT	There is no card in the reader.
SCARD_PRESENT	There is a card in the reader, but it has not been moved into position for use.
SCARD_SWALLOWED	There is a card in the reader in position for use. The card is not powered.
SCARD_POWERED	Power is being provided to the card, but the reader driver is unaware of the mode of the card.
SCARD_NEGOTIABLEMODE	The card has been reset and is awaiting PTS negotiation.
SCARD_SPECIFICMODE	The card has been reset and specific communication protocols have been established.

Value of dwPreferredProtocols	**Meaning**
SCARD_PROTOCOL_T0	Use the T=0 protocol.
SCARD_PROTOCOL_T1	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
SCARD_E_READER_UNAVAILABLE	- The reader has been removed.

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 if INFINITE is used. 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 will block for reader availability if cReaders is equal to zero and rgReaderStates is NULL.

```
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	**Meaning**
SCARD_STATE_UNAWARE	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.
SCARD_STATE_IGNORE	This reader should be ignored.
SCARD_STATE_CHANGED	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.
SCARD_STATE_UNKNOWN	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_A rgReaderStates[1];
LONG rv;

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

rgReaderStates[0].szReader         = strdup(“Reader X”);
rgReaderStates[0].dwCurrentState = SCARD_STATE_EMPTY;

cReaders = 1;
rv = SCardGetStatusChange( hContext, INFINITE, 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.
SCARD_E_READER_UNAVAILABLE - The reader is unavailable.

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 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.