MUSCLE PC/SC IFD Driver Developer Kit

IFD Driver Development Documentation

Written by David Corcoran corcoran@linuxnet.com
Last modified: August 1, 2000

Contents

MUSCLE PC/SC IFD DDK Documentation

Introduction / Overview

This document describes the API calls required to make a PC/SC driver for a device to be supported under the MUSCLE PC/SC resource manager. By implementing these calls correctly in a driver or shared object form, reader manufacturers can fit their hardware into an already existing infrastructure under several operating systems and hardware platforms. This IFD Handler interface is not restricted to smartcards and readers and could also be used for other types of smartcard like devices.

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 IFD DDK 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 pcsclite.h file.

```
DWORD                unsigned long
PDWORD               unsigned long *
UCHAR                unsigned char
PUCHAR               unsigned char *
LPSTR                char *
RESPONSECODE         long
VOID                 void
```

The following is a list of returns:

```
IFD_SUCCESS                          IFD_ERROR_TAG
IFD_ERROR_SET_FAILURE                IFD_ERROR_VALUE_READ_ONLY
IFD_ERROR_PTS_FAILURE                IFD_ERROR_NOT_SUPPORTED
IFD_PROTOCOL_NOT_SUPPORTED           IFD_ERROR_POWER_ACTION
IFD_ERROR_SWALLOW                    IFD_ERROR_EJECT
IFD_ERROR_CONFISCATE                 IFD_COMMUNICATION_ERROR
IFD_RESPONSE_TIMEOUT                 IFD_NOT_SUPPORTED
IFD_ICC_PRESENT                      IFD_ICC_NOT_PRESENT
```

Section 4

MUSCLE IFD DDK Routines

The routines specified hereafter will allow you to write an IFD handler for the PC/SC Lite resource manager. Please use the compliment developers kit complete with headers and Makefile at:

http://www.linuxnet.com/drivers/

This gives a common API for communication to most readers in a homogeneous fashion. This document assumes that the driver developer is experienced with standards such as ISO-7816-(1,2,3,4), EMV and MCT specifications. For listings of these specifications please access the above website.

MUSCLE PC/SC IFD DDK Documentation

Synopsis:

```
RESPONSECODE IFDHCreateChannel ( DWORD Lun, DWORD Channel );
```

Parameters:

Lun:	IN	Logical Unit Number.
Channel:	IN	Channel ID.

Description:

This function is required to open a communications channel to the port listed by Channel. For example, the first serial reader on COM1 would link to /dev/pcsc/1 which would be a sym link to /dev/ttyS0 on some machines This is used to help with intermachine independance.

On machines with no /dev directory the driver writer may choose to map their Channel to whatever they feel is appropriate.

Once the channel is opened the reader must be in a state in which it is possible to query IFDHICCPresence() for card status.

Lun - Logical Unit Number, use this for multiple card slots or multiple readers. 0xXXXXYYYY - XXXX multiple readers, YYYY multiple slots. The resource manager will set these automatically. By default the resource manager loads a new instance of the driver so if your reader does not have more than one smartcard slot then ignore the Lun in all the functions. Future versions of PC/SC might support loading multiple readers through one instance of the driver in which XXXX would be important to implement if you want this.

Channel - Channel ID. This is denoted by the following:
- 0x000001 - /dev/pcsc/1
- 0x000002 - /dev/pcsc/2
- 0x000003 - /dev/pcsc/3

USB readers can ignore the Channel parameter and query the USB bus for the particular reader by manufacturer and product id.

Returns:

IFD_SUCCESS	- Successful
IFD_COMMUNICATION_ERROR	- Error has occurred

MUSCLE PC/SC IFD DDK Documentation

Synopsis:

```
RESPONSECODE IFDHCloseChannel ( DWORD Lun );
```

Parameters:

Lun: IN Logical Unit Number.

Description:

This function should close the reader communication channel for the particular reader. Prior to closing the communication channel the reader should make sure the card is powered down and the terminal is also powered down.

Returns:

IFD_SUCCESS	- Successful
IFD_COMMUNICATION_ERROR	- Error has occurred

MUSCLE PC/SC IFD DDK Documentation

Synopsis:

```
RESPONSECODE IFDHGetCapabilities ( DWORD Lun, DWORD Tag,
                                   PDWORD Length, PUCHAR Value );
```

Parameters:

Lun:	IN	Logical Unit Number.
Tag:	IN	Tag of the desired data value.
Length:	INOUT	Length of the desired data value.
Value:	OUT	Value of the desired data.

Description:

This function should get the slot/card capabilities for a particular slot/card specified by Lun. Again, if you have only 1 card slot and don't mind loading a new driver for each reader then ignore Lun.

Tag - the tag for the information requested
Example: TAG_IFD_ATR - return the Atr and it's size (required).
these tags are defined in ifdhandler.h

Length - the length of the returned data
Value - the value of the data

Returns:

IFD_SUCCESS	- Success
IFD_ERROR_TAG	- Invalid tag given

MUSCLE PC/SC IFD DDK Documentation

Synopsis:

```
RESPONSECODE IFDHSetCapabilities ( DWORD Lun, DWORD Tag,
                                   DWORD Length, PUCHAR Value );
```

Parameters:

Lun:	IN	Logical Unit Number.
Tag:	IN	Tag of the desired data value.
Length:	INOUT	Length of the desired data value.
Value:	OUT	Value of the desired data.

Description:

This function should set the slot/card capabilities for a particular slot/card specified by Lun. Again, if you have only 1 card slot and don't mind loading a new driver for each reader then ignore Lun.

Tag - the tag for the information needing set.

Length - the length of the data
Value - the value of the data

Returns:

IFD_SUCCESS	- Success
IFD_ERROR_TAG	- Invalid tag given
IFD_ERROR_SET_FAILURE	- Could not set value
IFD_ERROR_VALUE_READ_ONLY	- Trying to set read only value

MUSCLE PC/SC IFD DDK Documentation

Synopsis:

```
RESPONSECODE IFDHSetProtocolParameters ( DWORD Lun, DWORD Protocol,
                                         UCHAR Flags, UCHAR PTS1,
                                         UCHAR PTS2, UCHAR PTS3 );
```

Parameters:

Lun:	IN	Logical Unit Number.
Protocol:	IN	Desired protocol.
Flags:	IN	OR'd Flags (See below).
PTS1:	IN	1st PTS Value.
PTS2:	IN	2nd PTS Value.
PTS3:	IN	3rd PTS Value.

Description:

This function should set the Protocol Type Selection (PTS) of a particular card/slot using the three PTS parameters sent

Protocol - 0 .... 14 T=0 .... T=14
Flags - Logical OR of possible values:
IFD_NEGOTIATE_PTS1 IFD_NEGOTIATE_PTS2 IFD_NEGOTIATE_PTS3
to determine which PTS values to negotiate.
PTS1,
PTS2,
PTS3 - PTS Values. (See ISO 7816 / EMV documentation)

Returns:

IFD_SUCCESS	- Success
IFD_ERROR_PTS_FAILURE	- Could not set PTS value
IFD_COMMUNICATION_ERROR	- Error has occurred
IFD_PROTOCOL_NOT_SUPPORTED	- Protocol is not supported

MUSCLE PC/SC IFD DDK Documentation

Synopsis:

```
RESPONSECODE IFDHPowerICC ( DWORD Lun, DWORD Action,
                            PUCHAR Atr, PDWORD AtrLength );
```

Parameters:

Lun:	IN	Logical Unit Number.
Action:	IN	Action to be taken.
Atr:	OUT	Answer to Reset (ATR) value of the inserted card.
AtrLength:	INOUT	Length of the ATR.

Description:

This function controls the power and reset signals of the smartcard reader at the particular reader/slot specified by Lun.

Action - Action to be taken on the card.

IFD_POWER_UP	Power and reset the card if not done so (store the ATR and return it and it's length).
IFD_POWER_DOWN	Power down the card if not done already (Atr/AtrLength should be zero'd)
IFD_RESET	Perform a quick reset on the card. If the card is not powered power up the card. (Store and return the Atr/Length)

Atr - Answer to Reset of the card. The driver is responsible for caching this value in case IFDHGetCapabilities is called requesting the ATR and it's length. The length should not exceed MAX_ATR_SIZE.

AtrLength - Length of the Atr. This should not exceed MAX_ATR_SIZE.

Notes:

Memory cards without an ATR should return IFD_SUCCESS on reset but the Atr should be zero'd and the length should be zero

Reset errors should return zero for the AtrLength and return IFD_ERROR_POWER_ACTION.

Returns:

IFD_SUCCESS	- Success
IFD_ERROR_POWER_ACTION	- Error powering/resetting card
IFD_COMMUNICATION_ERROR	- An error has occurred
IFD_NOT_SUPPORTED	- Action not supported

MUSCLE PC/SC IFD DDK Documentation

Synopsis:

```
RESPONSECODE IFDHTransmitToICC ( DWORD Lun, SCARD_IO_HEADER SendPci,
                                 PUCHAR TxBuffer, DWORD TxLength,
                                 PUCHAR RxBuffer, PDWORD RxLength,
                                 PSCARD_IO_HEADER RecvPci );
```

Parameters:

Lun:	IN	Logical Unit Number.
SendPci:	IN	Protocol structure
TxBuffer:	IN	APDU to be sent
TxLength:	IN	Length of sent APDU
RxBuffer:	OUT	APDU response
RxLength:	INOUT	Length of APDU response
RecvPci:	INOUT	Receive protocol structure

Description:

This function performs an APDU exchange with the card/slot specified by

Lun. The driver is responsible for performing any protocol specific exchanges such as T=0/1 ... differences. Calling this function will abstract all protocol differences.

SendPci contains two structure members:
Protocol - 0, 1, ... 14 (T=0 ... T=14)
Length - Not used.

TxBuffer - Transmit APDU example (0x00 0xA4 0x00 0x00 0x02 0x3F 0x00)
TxLength - Length of this buffer.
RxBuffer - Receive APDU example (0x61 0x14)
RxLength - Length of the received APDU. This function will be passed the size of the buffer of RxBuffer and this function is responsible for setting this to the length of the received APDU. This should be ZERO on all errors. The resource manager will take responsibility of zeroing out any temporary APDU buffers for security reasons.

RecvPci contains two structure members:
Protocol - 0, 1, ... 14 (T=0 ... T=14)
Length - Not used.

Notes:

The driver is responsible for knowing what type of card it has. If the current slot/card contains a memory card then this command should ignore the Protocol and use the MCT style commands for support for these style cards and transmit

them appropriately. If your reader does not support memory cards or you don't want to implement this functionality, then ignore this.

RxLength should be set to zero on error.

Returns:

IFD_SUCCESS	- Success
IFD_COMMUNICATION_ERROR	- An error has occurred
IFD_RESPONSE_TIMEOUT	- The response timed out
IFD_ICC_NOT_PRESENT	- ICC is not present
IFD_PROTOCOL_NOT_SUPPORTED	- Protocol is not supported

MUSCLE PC/SC IFD DDK Documentation

Synopsis:

```
RESPONSECODE IFDHControl ( DWORD Lun, PUCHAR TxBuffer,
                           DWORD TxLength, PUCHAR RxBuffer,
                           PDWORD RxLength );
```

Parameters:

Lun:	IN	Logical Unit Number.
TxBuffer:	IN	Bytes to be sent
TxLength:	IN	Length of sent bytes
RxBuffer:	OUT	Response
RxLength:	INOUT	Length of response

Description:

This function performs a data exchange with the reader (not the card) specified by Lun. Here XXXX will only be used.
It is responsible for abstracting functionality such as PIN pads, biometrics, LCD panels, etc. You should follow the MCT and CTBCS specifications for a list of accepted commands to implement. This function is fully voluntary and does not have to be implemented unless you want extended functionality.

TxBuffer - Transmit data
TxLength - Length of this buffer.
RxBuffer - Receive data
RxLength - Length of the received data. This function will be passed the length of the buffer RxBuffer and it must set this to the length of the received data.

Notes:

`RxLength should be zero on error.`

Returns:

IFD_SUCCESS	- Success
IFD_COMMUNICATION_ERROR	- An error has occurred
IFD_RESPONSE_TIMEOUT	- The response timed out

MUSCLE PC/SC IFD DDK Documentation

Synopsis:

```
RESPONSECODE IFDHICCPresence( DWORD Lun );
```

Parameters:

Lun: IN Logical Unit Number.

Description:

This function returns the status of the card inserted in the reader/slot specified by Lun.

Returns:

IFD_ICC_PRESENT	- ICC is present
IFD_ICC_NOT_PRESENT	- ICC is not present
IFD_COMMUNICATION_ERROR	- An error has occurred