NAME

Chipcard::PCSC::Card − Smart card communication library

SYNOPSIS

```
$hCard = new Chipcard::PCSC::Card ($hContext, "GemPC430 0 0",
       $Chipcard::PCSC::SCARD_SHARE_EXCLUSIVE);

$RecvData = $hCard−>Transmit([0xBC,0xB0,0x09,0xC8, 2]);

$hCard−>Disconnect($Chipcard::PCSC::SCARD_LEAVE_CARD);

$hCard−>Status();

$hCard−>BeginTransaction();

$hCard−>EndTransaction();

$hCard−>TransmitWithCheck($apdu, $sw_expected [, $debug]);

$hCard−>Control($control_code, \@data);

ISO7816Error($sw);
```

DESCRIPTION

The `Chipcard::PCSC::Card` module implements the `Chipcard::PCSC::Card` class. Objects from this class are used to communicate with a given reader. They are constructed out of a reference to a PCSC object that drives the reader.

For more information about PC/SC please read the ***pcscd (1)*** man page.

A `Chipcard::PCSC::Card` object uses the following property:

- **`$pcsccard_object`−>{hContext}**

  the reference to the underlying PCSC object

- **`$pcsccard_object`−>{hCard}**

  the current PCSC connection handle

- **`$pcsccard_object`−>{dwProtocol}**

  the protocol being used

CONSTRUCTORS

The following methods construct a `Chipcard::PCSC::Card` object:

- **`$hCard` = new Chipcard::PCSC::Card ($hContext);**

  Constructs a new `Chipcard::PCSC::Card` object without connecting to any reader.

  `$hContext` is mandatory and contains the reference to a valid PCSC object.

- **`$hCard` = new Chipcard::PCSC::Card ($hContext, `$reader_name`, `$share_mode`, `$preferred_protocol`);**

  Constructs a new Chipcard::PCSC::Card object and connect to the specified reader.

  - `$hContext`

    is mandatory and contains the reference to a valid PCSC object.

  - `$reader_name`

    is the name of the reader you want to connect to. It is of the form ‘‘GemPC410 0 0’’.

Please note that the list of available readers can be obtained with a call to `$hContext->ListReaders()`. (See the section named *PCSC METHODS* in the *Chipcard::PCSC* man page for more information on `ListReaders`).

- `$share_mode`

  is the desired mode of connection to the reader. It can be any of the following:

  - `$Chipcard::PCSC::SCARD_SHARE_EXCLUSIVE`

    the application do not share the reader

  - `$Chipcard::PCSC::SCARD_SHARE_SHARED`

    the application will allow others to share the reader.

  - `$Chipcard::PCSC::SCARD_SHARE_DIRECT`

    (not used by PC/SC−lite)

- `$preferred_protocol`

  is the protocol which should be used if possible. If the protocol is not available, another protocol will be used and `$hCard->{dwProtocol}` will be set accordingly. Both `$hCard->{dwProtocol}` and `$preferred_protocol` accept the following values:

  - `$Chipcard::PCSC::SCARD_PROTOCOL_T0`

    the T=0 protocol

  - `$Chipcard::PCSC::SCARD_PROTOCOL_T1`

    the T=1 protocol

  - `$Chipcard::PCSC::SCARD_PROTOCOL_RAW`

    raw protocol

- **`$hCard` = new Chipcard::PCSC::Card ($hContext, `$reader_name`, `$share_mode`);**

  This method is equivalent to:

  ```
   $hCard = new Chipcard::PCSC::Card ($hContext, $reader_name,
      $share_mode,
      $Chipcard::PCSC::SCARD_PROTOCOL_T0 |
      $Chipcard::PCSC::SCARD_PROTOCOL_T1);
  ```

- **`$hCard` = new Chipcard::PCSC::Card ($hContext, `$reader_name`);**

  This method is equivalent to:

  ```
   $hCard = new Chipcard::PCSC::Card ($hContext, $reader_name,
      $Chipcard::PCSC::SCARD_SHARE_EXCLUSIVE,
      $Chipcard::PCSC::SCARD_PROTOCOL_T0 |
      $Chipcard::PCSC::SCARD_PROTOCOL_T1);
  ```

CONSTRUCTION FAILURE

`Chipcard::PCSC::Card` constructors return an `undef` value when the object can not be created. `$Chipcard::PCSC::errno` can be used to get more information about the error. See section *ERROR HANDLING* in *Chipcard::PCSC* man page for more information.

Chipcard::PCSC::Card METHODS

Here is a list of all the methods that can be used with a `Chipcard::PCSC::Card` object.

`$hCard->`**Connect($reader_name,** `$share_mode`**,** `$preferred_protocol`**);**

`Connect()` can be used to connect to the reader and its smart card if the connection has not been established yet. The default constructor can establish the connection if given enough parameters.

The return value upon successful completion is the protocol used to communicate with the smart card. It can be any of the following:

- `$Chipcard::PCSC::SCARD_PROTOCOL_T0`

  the T=0 protocol
- `$Chipcard::PCSC::SCARD_PROTOCOL_T1`

  the T=1 protocol
- `$Chipcard::PCSC::SCARD_PROTOCOL_RAW`

  raw protocol
- `$reader_name`

  is mandatory. It contains the name of the reader you want to connect to. It is of the form ‘‘GemPC410 0 0’’.

  Please note that the list of available readers can be obtained with a call to `$hContext->ListReaders()`. (See the section named *PCSC METHODS* in the *Chipcard::PCSC* man page for more information on `ListReaders`).
- `$share_mode`

  is the desired mode of connection to the reader. It can be any of the following:

  - `$Chipcard::PCSC::SCARD_SHARE_EXCLUSIVE`

    the application do not share the reader
  - `$Chipcard::PCSC::SCARD_SHARE_SHARED`

    the application will allow others to share the reader.
  - `$Chipcard::PCSC::SCARD_SHARE_DIRECT`

    (not used by PCSClite)
- `$preferred_protocol`

  is the protocol which should be used if possible. If the protocol is not available, another protocol will be used and `$hCard->{dwProtocol}` will be set accordingly. `$preferred_protocol` accept the following values:

  - `$Chipcard::PCSC::SCARD_PROTOCOL_T0`

    the T=0 protocol
  - `$Chipcard::PCSC::SCARD_PROTOCOL_T1`

    the T=1 protocol
  - `$Chipcard::PCSC::SCARD_PROTOCOL_RAW`

    raw protocol

`$hCard->`**Connect($reader_name,** `$share_mode`**);**

This method is equivalent to:

```
$hCard->Connect($reader_name, $share_mode,
   $Chipcard::PCSC::SCARD_PROTOCOL_T0 |
   $Chipcard::PCSC::SCARD_PROTOCOL_T1);
```

`$hCard->`**Connect($reader_name);**

This method is equivalent to:

```
$hCard->Connect($reader_name, $Chipcard::PCSC::SCARD_SHARE_EXCLUSIVE,
   $Chipcard::PCSC::SCARD_PROTOCOL_T0 |
   $Chipcard::PCSC::SCARD_PROTOCOL_T1);
```

`$hCard->`**Reconnect($share_mode,** `$preferred_protocol`**,** `$initialization`**);**

`Reconnect()` can be used to re-negotiate an already opened connection. This implies that the `Chipcard::PCSC::Card` object is connected and has `$hCard->{hCard}` set accordingly.

Reconnecting to a smart card is used to change the share mode and the current protocol.

The return value upon successful completion is the protocol choose to communicate with the smart card. It can be any of the following:

- `$Chipcard::PCSC::SCARD_PROTOCOL_T0`

  the T=0 protocol
- `$Chipcard::PCSC::SCARD_PROTOCOL_T1`

  the T=1 protocol
- `$Chipcard::PCSC::SCARD_PROTOCOL_RAW`

  raw protocol
- `$share_mode`

  is the desired mode of connection to the reader. It can be any of the following:

  - `$Chipcard::PCSC::SCARD_SHARE_EXCLUSIVE`

    the application do not share the reader
  - `$Chipcard::PCSC::SCARD_SHARE_SHARED`

    the application will allow others to share the reader.
  - `$Chipcard::PCSC::SCARD_SHARE_DIRECT`

    (not used by PCSClite)
- `$preferred_protocol`

  is the protocol which should be used if possible. If the protocol is not available, another protocol will be used and `$hCard->{dwProtocol}` will be set accordingly. `$preferred_protocol` accept the following values:

  - `$Chipcard::PCSC::SCARD_PROTOCOL_T0`

    the T=0 protocol
  - `$Chipcard::PCSC::SCARD_PROTOCOL_T1`

    the T=1 protocol
  - `$Chipcard::PCSC::SCARD_PROTOCOL_RAW`

    raw protocol
- `$initialization`

  is the action to take when reconnecting to the smart card. It can be any of the following values:

  - `$Chipcard::PCSC::SCARD_LEAVE_CARD`

    do nothing on close
  - `$Chipcard::PCSC::SCARD_RESET_CARD`

    reset on close
  - `$Chipcard::PCSC::SCARD_UNPOWER_CARD`

    power down on close
  - `$Chipcard::PCSC::SCARD_EJECT_CARD`

eject on close

`$hCard->`**Reconnect($share_mode,** `$preferred_protocol`**);**

This method is equivalent to:

```
$hCard->Reconnect($share_mode, $preferred_protocol,
   $Chipcard::PCSC::SCARD_LEAVE_CARD);
```

`$hCard->`**Reconnect($share_mode);**

This method is equivalent to:

```
$hCard->Reconnect($share_mode,
   $Chipcard::PCSC::SCARD_PROTOCOL_T0 |
   $Chipcard::PCSC::SCARD_PROTOCOL_T1,
   $Chipcard::PCSC::SCARD_LEAVE_CARD);
```

`$hCard->`**Reconnect();**

This method is equivalent to:

```
$hCard->Reconnect($Chipcard::PCSC::SCARD_SHARE_EXCLUSIVE,
   $Chipcard::PCSC::SCARD_PROTOCOL_T0 |
   $Chipcard::PCSC::SCARD_PROTOCOL_T1,
   $Chipcard::PCSC::SCARD_LEAVE_CARD);
```

`$hCard->`**Disconnect($initialization);**

`Disconnect()` closes the connection to the smart card reader. It returns true upon successful completion or undef otherwise. `$hCard->{hContext}` will be set to undef if the connection is successfully closed.

- `$initialization`

  is the action to take when reconnecting to the smart card. It can be any of the following values:

  - `$Chipcard::PCSC::SCARD_LEAVE_CARD`

    do nothing on close

  - `$Chipcard::PCSC::SCARD_RESET_CARD`

    reset on close

  - `$Chipcard::PCSC::SCARD_UNPOWER_CARD`

    power down on close

  - `$Chipcard::PCSC::SCARD_EJECT_CARD`

    eject on close

`$hCard->`**Disconnect();**

This method is equivalent to:

```
$hCard->Disconnect($Chipcard::PCSC::SCARD_EJECT_CARD);
```

`$hCard->`**Status();**

`Status()` returns the current status of the connection to a smart card. It is used to retrieve the ATR (Answer To Reset) value as well as the protocol being used to communicate.

The return value is the `undef` value if an error occurs. In such a case, `$!` should be set with a string describing the error. Upon successful completion `Status` returns an array as follows: (`$reader_name`, `$reader_state`, `$protocol`, `\@atr`)

- `$reader_name`

  is a string containing the name of the reader

- `$reader_state`

  is a scalar containing the current state of the reader.

It can be any combination of the following values:

- `$Chipcard::PCSC::SCARD_UNKNOWN`

  unknown state

- `$Chipcard::PCSC::SCARD_ABSENT`

  card is absent

- `$Chipcard::PCSC::SCARD_PRESENT`

  card is present

- `$Chipcard::PCSC::SCARD_SWALLOWED`

  card not powered

- `$Chipcard::PCSC::SCARD_POWERED`

  card is powered

- `$Chipcard::PCSC::SCARD_NEGOTIABLE`

  ready for PTS

- `$Chipcard::PCSC::SCARD_SPECIFIC`

  PTS has been set

- `$protocol`

  is the protocol being used. It can be any of the following values:

  - `$Chipcard::PCSC::SCARD_PROTOCOL_T0`,
  - `$Chipcard::PCSC::SCARD_PROTOCOL_T1`,
  - `$Chipcard::PCSC::SCARD_PROTOCOL_RAW`

- \@atr

  is a reference to an array containing the ATR. Each cell of the array contains one byte of the ATR. This parameter is however optional as the ATR may not be available under some circumstances. For instance when the card is not inserted, no ATR can be returned and this parameter will be `undef`.

`$hCard->`**Transmit(\@data);**

`Transmit()` is used to exchange data with the card.

It returns a reference to an anonymous array holding the answer to the emitted data. In case of an error, the reference is the `undef` value.

- \@data

  is a reference to the data to be sent to the card.

Here is a small sample of how to use `transmit`:

```
 $SendData = [0x00, 0xA4, 0x01, 0x00, 0x02, 0x01, 0x00];
 $RecvData = $hCard->Transmit($SendData);

 print "  Recv = ";
 foreach $tmpVal (@{$RecvData}) {
     printf ("%02X ", $tmpVal);
 } print "\n";
```

`$hCard->`**BeginTransaction();**

`BeginTransaction()` is used to temporarily get exclusive control over the smart card.

It returns TRUE upon successful completion and FALSE otherwise. `$Chipcard::PCSC::errno` should be set accordingly in case of an error. See section *ERROR HANDLING* in *Chipcard::PCSC* man page for more information.

`$hCard->`**EndTransaction($disposition);**

`EndTransaction()` is used to end a transaction initiated with `BeginTransaction()`.

It returns `TRUE` upon successful completion and FALSE otherwise. `$Chipcard::PCSC::errno` should be set accordingly in case of an error. See section *ERROR HANDLING* in *Chipcard::PCSC* man page for more information.

- `$disposition`

  is the action to take when ending the transaction. It can be any of the following values:

  - `$Chipcard::PCSC::SCARD_LEAVE_CARD`

    do nothing on close

  - `$Chipcard::PCSC::SCARD_RESET_CARD`

    reset on close

  - `$Chipcard::PCSC::SCARD_UNPOWER_CARD`

    power down on close

  - `$Chipcard::PCSC::SCARD_EJECT_CARD`

    eject on close

`$hCard->`**EndTransaction();**

This method is equivalent to:

```
 $hCard->EndTransaction($Chipcard::PCSC::SCARD_LEAVE_CARD);
```

`$hCard->`**TransmitWithCheck($apdu,** `$sw_expected` **[,** `$debug`**]);**

This method is a wrapper around `$hCard->`**Transmit()**. The `$apdu` parameter is an ASCII text like “00 A4 01 00 02 01 00”, `$sw_expected` is a Perl regular expression like “90 [01]0”.

If the status word returned matches the expression `$sw_expected` the method returns a list ($sw, `$recv`). `$sw` is the status word (like “90 00”) of the command, `$recv` is the result of the command.

If the status word do not match the expression `$sw_expected` the method returns undef and the variable `$Chipcard::PCSC::Card::Error` is set.

The `$debug` argument is optional. If present the method will print on stdout the command sent and the response from the card.

Example:

```
 ($sw, $RecvData) = $hCard->TransmitWithCheck($SendData, "6E 00", 1);
 warn "TransmitWithCheck: $Chipcard::PCSC::Card::Error" unless defined $sw;
```

`$hCard->`**Control($control_code, \@data);**

This method uses PC/SC **SCardControl()** to send data specific to the reader driver. See your driver documentation to know what data to use.

Example:

```
 $data = Chipcard::PCSC::ascii_to_array ("01 23 45");
 $RecvData = $hCard->Control(0x42000001, $SendData);
 die ("Can't Control data: $Chipcard::PCSC::errno") unless (defined ($RecvData));
```

ISO7816Error($sw);

This method returns the ASCII text corresponding to the status word `$sw` according to ISO 7816−4 specifications.

Example:

```
       $sw = "90 00";
       print "$sw: " . &Chipcard::PCSC::Card::ISO7816Error($sw) . "\n";
```

SEE ALSO

pcscd man page has useful information about PC/SC−lite. *Chipcard::PCSC* man page holds all the necessary information to create the PCSC object which will be the basis of `Chipcard::PCSC::Card`.

COPYRIGHT

(C) Lionel VICTOR, 2001, GNU GPL

(C) Ludovic ROUSSEAU, 2003−2008, GNU GPL

AUTHORS / ACKNOWLEDGEMENT

```
       Lionel VICTOR <lionel.victor@unforgettable.com>
                     <lionel.victor@free.fr>

       Ludovic ROUSSEAU <ludovic.rousseau@free.fr>
```