Thank You for your interest to the SCardX Easy smart card ActiveX control!

Please send me all your suggestions or any questions about the SCardX Easy smart card ActiveX control via e-mail igor@scardsoft.com.

Visit our web site for the latest software and specifications updates.

Yours,
Igor V. Kharchenko
author.
# Table of Contents

## Part I  About

1. About SCardX Easy ActiveX control ................................................................. 4
2. Contacts ........................................................................................................ 4

## Part II  SCardX Easy ActiveX control overview

1. What is the SCardX Easy? .............................................................................. 5
   - SCardX Easy is an ActiveX ...................................................................... 5
   - What SCardX Easy can to add into your web page? ............................... 5
   - Smart cards on your web pages .............................................................. 5
2. Appearance ................................................................................................... 6
   - States page ................................................................................................ 6
   - Events History page .................................................................................. 8
   - ToolBar panel ............................................................................................ 8
   - StatusBar panel .......................................................................................... 9
3. Smart card functionality ............................................................................... 10
   - Smart card service ................................................................................... 10
   - Events ....................................................................................................... 10
   - Data sending ............................................................................................. 11
4. Additional tools ............................................................................................ 11
   - LookUp service ......................................................................................... 11
   - Data ciphering .......................................................................................... 12
   - Tray Icon usage ....................................................................................... 12
   - Preferences ............................................................................................... 12

## Part III  SCardX Easy first start

1. Adding the SCardX Easy ActiveX control to the web page .......................... 14
2. Your first smart card web page and connection test ..................................... 17

## Part IV  Your first smart card web page. "Hello, cards World !"

1. Demo web page ............................................................................................. 21
2. New web page ............................................................................................... 22
3. Interface procedures .................................................................................... 23
4. Events ......................................................................................................... 24
5. Preparing the connection controls ............................................................... 25
6. Preparing the opened reader controls ......................................................... 27
7. Tray Icon ..................................................................................................... 31
8. LookUp service ........................................................................................... 35
9. Data ciphering ............................................................................................. 35
Part V SCardX Easy interface specification

1 Properties

ActivePage ................................................................. 42
BorderStyle .............................................................. 44
BorderWidth ............................................................. 44
ConnectionState ......................................................... 46
EventsHistoryEnabled ............................................ 46
EventsLogging ........................................................ 47
SeparateReceivedBytes ........................................... 47
SmartCardService .................................................... 48
Visible ................................................................. 49
VisibleEventsHistory ............................................... 49
VisibleStatus Bar ...................................................... 50
VisibleToolBar ......................................................... 50
VisibleTrayIcon ........................................................ 51

2 Functions

DES_DecryptString ................................................... 52
DES_EncryptString ................................................... 53
EventsHistoryClear .................................................. 55
Finalize ............................................................... 55
GetCardATR ............................................................ 55
GetCardInfo ............................................................ 57
GetCardInfoFmt ....................................................... 57
GetEventsHistory ..................................................... 58
GetReaderInfo ......................................................... 59
GetReaderInfoFmt ..................................................... 61
GetReadersList ........................................................ 61
IsCardReady ........................................................... 62
IsLocked ............................................................... 63
LookUpError .......................................................... 63
LookUpReaderState .................................................. 64
ReopenReader .......................................................... 64
SendCardAPDU ....................................................... 65
SendCardDATA ........................................................ 68
SetPref_PCSOnCardDetect ....................................... 68
TrayIconMenuClear ................................................... 70
TrayIconMenuCreate ............................................... 71
TrayIconMenultemSetChecked ................................... 72
TrayIconMenultemSetDefault .................................... 73
TrayIconMenultemSetEnabled .................................... 74
Version ............................................................... 75
VersionMajor .......................................................... 75
VersionMinor .......................................................... 76

3 Events

OnCardDetected ....................................................... 77
OnCardInvalid ........................................................ 78
OnCardReady .......................................................... 79
OnCardWait ........................................................... 80
OnConnected .......................................................... 81
## Part VI Registration

1. **Unregistered version limitations** .................................................. 90
2. **Licensing** ................................................................................. 90
3. **Registration steps** ..................................................................... 93

### 1 Unregistered version limitations

- **Step 1** : License Query ................................................................. 93
- **Step 2** : Purchasing the License ................................................ 93
- **Step 3** : Certificate registration ................................................ 93
1 About

1.1 About SCardX Easy ActiveX control

SCardX Easy
Smart Card ActiveX control
Version 1.3

Copyright © 2005 by SCardSOFT

1.2 Contacts

The official web site of SCardX Easy is the SCardSOFT homepage:

useful SCardSOFT pages:

SCardX Easy official web page
SCardSOFT Home
Smart Cards specifications Library page
Smart Cards Forum (English)
Smart Cards Forum (Russian)
Prices page
License's purchasing info page

contact e-mail addresses:

info@scardsoft.com - common questions;
sales@scardsoft.com - payments and licenses questions;
support@scardsoft.com - support service;
2 SCardX Easy ActiveX control overview

2.1 What is the SCardX Easy?

2.1.1 SCardX Easy is an ActiveX

SCardX Easy is a standart ActiveX control.

If your development environment (IDE) supports the ActiveX technology like the MS Visual Studio, Borland Delphi or C++ Builder or other - than the SCardX Easy may be successfully used by your applications.

You can use SCardX Easy for working with any smart card on any your web page. Only one limitation present today: - you must open your smart card web pages by the Microsoft Internet Explorer web browser only.

2.1.2 What SCardX Easy can to add into your web page?

SCardX Easy adds to your web page the following functionality:

smart cards functionality:
- receiving the smart card service's and devices' events;
- receiving an information about the attached devices;
- receiving an information about the opened smart card;
- sending the command data buffers into the opened smart cards and receiving the cards responses;
- managing the cards opening and closing modes;

additional useful tools:
- Error LookUp and Reader States LookUp services
- Data ciphering
- Tray Icon usage

2.1.3 Smart cards on your web pages

The SCardX Easy ActiveX control creates the communication channel between the parent application (web page) and an opened smart card via the smart card service and any attached PC/SC compatible smart card reader.

The SCardX Easy allows you to send the command data buffers into any ISO-7816 compatible smart cards and to receive the cards' answers.

Using SCardX Easy ActiveX control you can talk with a smart card using card's "native" language - the language of the command APDU's. It is the lowest level of work with smart cards from the PC.

Using SCardX Easy ActiveX control you can send into your cards any commands according to the cards' specifications easy and without any limitations.
2.2 Appearance

2.2.1 States page

The "States" page is a main user interface element of the SCardX Easy ActiveX control.

There are many useful information and context pop-up menu commands on this page:

**Smart card service info:**
- selected smart card service
- service connection state

**Your License info:**
- License owner's name and address
- License number
- License type
- License usage rules

**Preferences:**

PC/SC Card detecting defaults
- Open the reader automatically: Yes, No
- Preferred Protocol: T0, T1, RAW, Autodetect, Undefined
- Preferred Sharing Mode: Share reader, Exclusive use, Direct reader control
- Card closing mode: Live card, Reset card, Unpower card, Eject card

Miscellaneous
- Separate received HEX bytes: Yes, No
- Events logging: Log all events, Log most useful events only

**Attached devices' list:**
- Device state
- Device info
Opened smart card info:

- ATR
- Protocol
- Sharing mode
- Card info

Error

- The last error info

This page has the context pop-up menu which allows you to take access to many useful commands depending to the selected item.
2.2.2 Events History page

This page contents the archive of the events which was occured.

<table>
<thead>
<tr>
<th>N</th>
<th>Source</th>
<th>Event</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>MS Smart Card service</td>
<td>Service connected</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>AKS id# 0</td>
<td>Reader state changed</td>
<td>0x00000012: There</td>
</tr>
<tr>
<td>4</td>
<td>AKS id# 1</td>
<td>Reader state changed</td>
<td>0x00000012: There</td>
</tr>
<tr>
<td>5</td>
<td>SCM Microsystems Inc. CHIDFI</td>
<td>Reader state changed</td>
<td>0x0000C012: There</td>
</tr>
<tr>
<td>6</td>
<td>SCM Microsystems Inc. CHIDFI</td>
<td>Waiting for card</td>
<td>Insert card into a rec</td>
</tr>
<tr>
<td>7</td>
<td>AKS id# 1</td>
<td>Waiting for card</td>
<td>Insert card into a rec</td>
</tr>
<tr>
<td>8</td>
<td>AKS id# 0</td>
<td>Waiting for card</td>
<td>Insert card into a rec</td>
</tr>
<tr>
<td>9</td>
<td>SCM Microsystems Inc. CHIDFI</td>
<td>Reader state changed</td>
<td>0x0000D0022: There</td>
</tr>
<tr>
<td>10</td>
<td>SCM Microsystems Inc. CHIDFI</td>
<td>Card detected</td>
<td>Card was detected in</td>
</tr>
<tr>
<td>11</td>
<td>SCM Microsystems Inc. CHIDFI</td>
<td>Reader state changed</td>
<td>0x0000D0122: There</td>
</tr>
<tr>
<td>12</td>
<td>SCM Microsystems Inc. CHIDFI</td>
<td>Card ready</td>
<td>ATR = 38 79 94 00</td>
</tr>
</tbody>
</table>

**Fields**
- N: the serial number of the event;
- Source: event source;
- Event: event message;
- Value: event value (if present);
- Event Time: the time when the event was occured;

**Pop-up Menu Commands**
- First Event: go to a first record;
- Last Event: go to a last record;
- Events logging: the logging mode: Log all events, Log most useful events only
- Save Events History: save grid data to a text file;
- Copy Events History: copy grid data to a Windows Clipboard;
- Clear All: clear all events messages at once;

**Useful info:**
- you can hide/show this page by operating of the VisibleEventsHistory property;
- you can read the Events History grid data to your web page by calling the function GetEventsHistory;
- you can clear the Events History grid data by calling the function EventsHistoryClear;
- you can lock/unlock the events logging by operating of the EventsHistoryEnabled property.

2.2.3 ToolBar panel

The ToolBar panel contain the controls for the data sending.

Using the ToolBar you can prepare and send into an opened smart card the control APDU's.
Or you can prepare and send into an opened smart card the unformatted data buffers.

The ToolBar may be used for testing of the smart card service connection or your device from any temporary web page because it is ready for data sending at once after adding the SCardX Easy to your web page.

If you don’t need the ToolBar into your web page you can hide it easy.

Useful info:
- you can hide/show the ToolBar by operating of the VisibleToolBar property;

### 2.2.4 StatusBar panel

The StatusBar is an indicator of the activity of the data exchange process between the SCardX Easy and a smart card service.

If the control is locked the Led is On.
2.3 Smart card functionality

2.3.1 Smart card service

The smart card service is a drivers’ layer which is used by SCardX Easy for communication with a smart card.

Each card readers’ manufacturer supports its devices by its own drivers’ set.

However, the last versions of the Microsoft Windows OS supports its own smart card service based on the PC/SC standard. The Microsoft PC/SC smart card service allows any applications to work with smart cards independent of the hardware drivers.

Today, SCardX Easy supports the MS Smart Card Service (PC/SC Interface) and it works with any of PC/SC compatible smart card readers.

The next versions of SCardX Easy will additionally support some another alternative smart card services.

Useful info:

- you can select the smart card service by operating of the SmartCardService property;
- you can connect SCardX Easy to the selected service or disconnect it by operating of the ConnectionState property;

2.3.2 Events

The SCardX Easy allows to your web page to receive all possible events from the selected smart card service:

User interface events
OnHistoryEvent
OnReaderSelected
OnTrayIconDb1Click
OnTrayIconMenu1tem

Smart card work events

OnCardDetected
OnCardInvalid
OnCardReady
OnCardWait
OnConnected
OnDataSent
OnDisconnected
OnReadersList
OnReaderStateChanged

Other events

OnERROR
OnLock
OnUnlock

2.3.3 Data sending

The SCardX Easy allows to your web page to send the data into a card and to receive the card answers.

The data sending functions are:

- **SendCardAPDU**: sending the command APDU’s;
- **SendCardDATA**: sending the unformatted data buffers;

Before the data sending your web page must prepare the sending data in the hexadecimal format according to the specification of your card.

After calling both these functions returns the hexadecimal data buffer of the card answer on the sent data.

You may analyze the card answers according to the cards’ specifications.

2.4 Additional tools

2.4.1 LookUp service

The SCardX Easy allows to your web page to use the following LookUp services:

- **Error LookUp**: decodes any error code from it number value to the text string;
- **State LookUp**: decodes and unpacks the readers’ state code from it number value to the text string;
2.4.2 Data ciphering

The SCardX Easy allows to your web page to encode and to decode the text strings using the DES algorithm.

The DES ciphering functions are:

- `DES_EncryptString`: for encrypting the text;
- `DES_DecryptString`: for decrypting text from an encrypted hex data buffer;

2.4.3 Tray Icon usage

The SCardX Easy has its own icon in the system tray zone.

By default this icon has a single pop-up menu item "About...".

You can expand this pop-up menu by adding of your own menu items at any time.

The SCardX Easy allows you to add any counts of your own menu items.

**Useful info:**

- you can re-create the TrayIcon's menu by calling the `TrayIconMenuCreate` function;
- you can clear all menu items of the TrayIcon at once by calling the `TrayIconMenuClear` function;
- you can check/uncheck the menu item by calling the `TrayIconMenuItemSetChecked` function;
- you can enable/disable the menu item by calling the `TrayIconMenuItemSetEnabled` function;
- you can make the menu item as a default item by calling the `TrayIconMenuItemSetDefault` function;
- when the user clicks on the TrayIcon menu item the event `OnTrayIconMenuItem` occurs;
- when the user twice clicks on the TrayIcon the event `OnTrayIconDblClick` occurs;

2.4.4 Preferences

The SCardX Easy allows you to change the preferences via its ActiveX interface.

**PC/SC Card detecting defaults**

Using the `SetPref_PCSC_OnCardDetect` function you can set up of the following preferences:
- Open the reader automatically
- Preferred Protocol
- Preferred Sharing Mode
- Card closing mode

**Miscellaneous**

Using the `SeparateReceivedBytes` property you can set up the "Separate received HEX bytes" parameter of the control's preferences.

Using the `EventsLogging` property you can set up the "Events logging" parameter of the control's preferences.
3 SCardX Easy first start

3.1 Adding the SCardX Easy ActiveX control to the web page

Create the new web page using an any html editor:

```html
<HTML>
<HEAD>
<META name=VI60 defaultClientScript content=JavaScript>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<TITLE></TITLE>
</HEAD>

<BODY>
</BODY>
</HTML>
```

Add the following text into this new html file:

```html
<OBJECT
   CLASSID="clsid:25F6377F-63FC-4741-891B-2DDAD6DD11DA"
   id=SCardX_Easy
>
</OBJECT>
```
If you want to use the SCardX Easy ActiveX control in the invisible mode please put this OBJECT tag in the HEAD section of your HTML file:

```
<Object
   CLASSID="clsid:25F6577F-63FC-4741-891E-2B0AD6DD11DA"
   id="SCardX_Easy"
>
</Object>
```

That's all. You can use now the SCardX Easy ActiveX control on this web page:
3.2 Your first smart card web page and connection test

You can create the simple web page for testing of the smart card service and card readers of your PC.

Please create the new JavaScript web page and add the SCardX Easy ActiveX control into its BODY section:

```html
<OBJECT CLASSID="clsid:25F5377F-63FC-4741-891B-2DEAD6DD11BA"
        id="SCardX_Easy">
</OBJECT>
</BODY>
</HTML>
```

Setting up the page startup and page closing

Please add the following text in the HEAD section of your HTML file:

```html
<SCRIPT ID=clientEventHandlersJS LANGUAGE=javascript>
<!--
function  window_onload() {
  SCardX_Easy.SmartCardService = 0;
  SCardX_Easy.ConnectionState = 0;
}

function  window_onunload() {
  SCardX_Easy.Finalize();
}
//-->
</SCRIPT>
```

And modify the BODY tag of your HTML file:
Open this web page in the MS Internet Explorer.

Click on the "Service" item of the "States" page of the SCardX Easy by the right mouse button and select the menu item "Connect ":

The SCardX Easy ActiveX control will try to connect the MS Smart Card service.

If these drivers are present on your PC the SCardX Easy ActiveX control will connect its and the available card readers names will be shown on the "States" page.
Insert the standard ISO-7816 smart card like the GSM SIM into the reader.

Warning! Do not use the memory cards for this test!

If the card is valid it will be opened and the info about of this card will be shown on the “States” page. Click on the highlighted reader item.

Open the “Events History” page.
Click on the "Send APDU" button of the ToolBar panel.

If the data will be sent into the card correctly:

- the event "Data sent" will be occurred and placed into the events history grid;
- the received card answer will be placed into the "Received data" controls of the ToolBar panel;

Otherwise an error event will be created and placed into the events history grid.

That's all.

If you can send the data buffers into your cards you may start now to create your first smart card web page.

If an error event will be occurred during of this test it means that either the smart card service on the your PC is not started or your devices are not works. In this case you can contact the SCard SOFT's support service via e-mail support@scardsoft.com for assisting in detecting and removing the troubles.
4 Your first smart card web page. "Hello, cards World!"

4.1 Demo web page

The SCardX Easy setup program installs the source codes of example applications.

The default examples’ path on your hard drive after the control’s installation is:

"C:\Program Files\SCardSOFT\SCardX Easy\- Examples"

You can find the JavaScript demo web page on the following default path:

"C:\Program Files\SCardSOFT\SCardX Easy\- Examples\IEExplorer JavaScript"

The JavaScript demo web page looks like on this picture:

![JavaScript demo web page](image)

This JavaScript demo web page will be used by this Manual as a base of your first smart card web page.

Please find it and copy its source HTML file to your JavaScript projects workplace.
4.2 New web page

1. Create the new web page;
2. Add all standard web page controls which you need;
3. Add the SCardX Easy into a created html page;
4. Rename the SCardX Easy object to SCardX_Easy;
5. Set up the control’s position on the page.
4.3 Interface procedures

You need to control the states of the web page controls depending on the states of the connection and to your readers' states.

For example, the data sending button must be disabled while the reader is empty.

For managing of the web page controls' states you need to control the values of the following three SCardX Easy ActiveX control properties:

- ConnectionState
- IsLocked
- IsCardReady

When one of these properties becomes changed you need enable or disable some of the web page controls.

The demo web page has two interface functions:

```javascript
function EnableControls()
{
  Connected = (SCardX_Easy.ConnectionState == 1);
  RList.disabled = (!Connected);
  CommandConnect.disabled = (Connected);
  CommandDisconnect.disabled = (!Connected);
  CommandRefresh.disabled = (!Connected);
  EnableCardReadyControls();
}

function EnableCardReadyControls()
{
  RName = RList.value;
  CardReady = SCardX_Easy.IsCardReady( RName );
  CommandRInfo.disabled = (!CardReady);
  CommandCInfo.disabled = (!CardReady);
  CommandATR.disabled = (!CardReady);
  CommandReopen.disabled = (!CardReady);
  CommandAPDU.disabled = (!CardReady);
  CommandDATA.disabled = (!CardReady);
}
```

Call the function EnableControls on the following events:

- OnConnected
- OnDisconnected
- OnLock
- OnUnlock

The EnableControls function calls the EnableCardReadyControls function automatically. But you need to call it additionally on the following events:

- OnCardWait
- OnCardReady
- OnReaderSelected

And additionally you need to call the EnableCardReadyControls function each time when the active reader name of your web page will be changed. In the demo web page this function additionally calls on the RList_onchange event.
4.4 Events

How to receive the smart card readers' events into your web page?

It's so easy by using of the SCardX Easy ActiveX control!

All SCardX Easy ActiveX control events are maximal informative.

For example **OnCardReady** event:

```
<SCRIPT
    LANGUAGE=javascript
    FOR=SCardX_Easy
    EVENT="OnCardReady\{ReaderName,ATR,ProtocolValue,Protocol\}"
>
<!--
    ' .... your code here
//-->
</SCRIPT>
```

The **OnCardReady** event gives you all useful information about the opened smart card at once:
- the opened card's **Reader Name**;
- the **ATR** of the opened card;
- the real active **Protocol** of the opened card.

Any smart card application or web page without the events are dead and unusable.

Otherwise by using the SCardX Easy ActiveX control you can add to your web pages the power and sensitivity of the professional smart card applications.

**Processing of the received events**

Each event has its own parameters list and each event is intended for its own task.

Additionally to the specific events' tasks the demo web page has special controls and functions for the simple visualization of all received events:

![Received ActiveX Events](image)

These events visualization tools are the EventsList textarea control and the AddMessage function:

```javascript
function AddMessage(EventStr, EventSource){
    if (EventSource!="") { dvd= " : ";} else {dvd="";};
    var bbb=EventsList.value;
    var aaa= ">>> " + EventStr + dvd + EventSource + strNewLine;
    EventsList.value=aaa+bbb;
}
```
It's easy! Call this function for each occurred event and you will see all received events by looking thru the text lines into the EventsList control:

```javascript
<SCRIPT LANGUAGE=javascript FOR=SCardX_Easy EVENT="OnReaderSelected( ReaderName )">
<!---
AddMessage( "OnReaderSelected", ReaderName );
//-->
</SCRIPT>
```

### 4.5 Preparing the connection controls

Before working with smart cards we need to connect the smart card service.

Place on the web page the three buttons for the connection commands and one select control for the readers names’ list like on this picture:

![Buttons and Select Reader](image)

#### Service connecting commands

By clicking on the "Connect" button we’ll select the MS Smart Card service and set up the connection state of SCardX Easy as connected:

```javascript
function CommandConnect_onclick () {
    SCardX_Easy.SmartCardService = 1;
    SCardX_Easy.ConnectionState = 1;
}
```

By clicking on the "Disconnect" button we’ll close the connection and unload the driver:

```javascript
function CommandDisconnect_onclick () {
    SCardX_Easy.ConnectionState = 0;
    SCardX_Easy.SmartCardService = 0;
}
```

What will be happened after clicking on the "Connect Service" button:

- the SCardX Easy loads the driver libraries and makes the connection to the selected smart card service;
- OnConnected events occurs; on this event you can enable the controls on the web page by calling the EnableControls function;
- the SCardX Easy loads from the service the list of the names of the available card readers which are attached to your PC;
- OnReadersList events occurs; on this event you can receive and store the readers list;
- the SCardX Easy starts to listen the devices for the changes of its states;
from now the web page will receive the following readers states events:

- **OnReaderStateChanged**
- **OnCardWait**: on this event you can enable the controls on the web page by calling the `EnableCardReadyControls` function;
- **OnCardDetected**
- **OnCardInvalid**: on this event you can enable the controls on the web page by calling the `EnableCardReadyControls` function;
- **OnCardReady**: on this event you can enable the controls on the web page by calling the `EnableCardReadyControls` function;
- **OnReaderSelected**: on this event you can enable the controls on the web page by calling the `EnableCardReadyControls` function;

### Readers list receiving

Many functions of the SCardX Easy ActiveX control needs the reader name as a parameter.

You can receive and store on the web page the readers list by the two ways:

- using the **OnReadersList** event;
- using the **GetReadersList** function of the SCardX Easy;

The demo web page uses the RList select control as a readers names’ container. And additionally the selected reader of this control always used as the active reader name for all smart cards’ and devices’ commands.

For filling up of the RList select control by the real names of attached readers the demo web page has a function `MakeReadersList`:

```javascript
function MakeReadersList(ReadersNames){
    var ii=0;
    var part_num=0;
    var ReadersNamesList = new Array();
    var ss="";
    RList.length=0;
    ReadersNamesList = ReadersNames.split(strNewLine);
    while (part_num < ReadersNamesList.length)
    {
        ss=ReadersNamesList[part_num];
        ss=ss.replace(/\s*|\s*/g,"");
        if (ss>""){
            RList.length=part_num+1;
            RList.options[part_num].text=ss;
            RList.options[part_num].value=ss;
        }
        part_num+=1;
    }
    RList.options[0].selected=true;
}
```

The demo web page calls the `MakeReadersList` automatically on the **OnReadersList** event:

```html
<SCRIPT LANGUAGE=javascript FOR=SCardX_Easy EVENT="OnReadersList(ReadersList)">
<!--
AddMessage("OnReadersList", ");
if (ReadersList>"") { MakeReadersList(ReadersList); }
//-->
</SCRIPT>
```

It's easy! The SCardX Easy ActiveX control gives you the readers list as a parameter of the **OnReadersList** event!

Alternatively you can receive the readers list at any time using the **GetReadersList** function of the SCardX Easy. For this command the demo web page has the "Refresh Readers List" button.

By clicking on the "Refresh Readers List" button the web page reloads the readers list:
function CommandRefresh_onclick () {
  ReadersList=SCardX_Easy.GetReadersList ();
  if (ReadersList!="") {
    MakeReadersList (ReadersList );
  }
}

4.6 Preparing the opened reader controls

After receiving of the OnCardReady event the web page may call the following functions of the SCardX Easy ActiveX control:

- ReopenReader
- GetReaderInfoFmt
- GetReaderInfo
- GetCardInfoFmt
- GetCardInfo
- GetCardATR
- SendCardAPDU
- SendCardDATA

All these functions needs the opened reader name as a parameter and may be called after receiving the OnCardReady event only.

**Reopen Reader command**

Add the "Reopen Reader" button on the web page.

By clicking on this button the web page will reopens the selected card reader:

function CommandReopen_onclick () {
  SCardX_Easy.ReopenReader (RList.value);
}

**Receiving the Reader Info**

Add the "Get Reader Info" button and the "Format Info" checkbox on the web page.
The SCardX Easy has two functions for the reader info receiving:

- `GetReaderInfo`
- `GetReaderInfoFmt`

The function `GetReaderInfo` returns the not formatted info lines like these ones:

```
[VENDOR INFO]
VENDOR NAME=SCM Microsystems Inc.
VENDOR IFD TYPE=CHIPDRIVE Serial
VENDOR IFD VERSION=< no info >
VENDOR IFD SERIAL NO=12639860
```

The function `GetReaderInfoFmt` returns the formatted info lines like these ones:

```
VENDOR INFO
VENDOR NAME .................. SCM Microsystems Inc.
VENDOR IFD TYPE .............. CHIPDRIVE Serial
VENDOR IFD VERSION ........... < no info >
VENDOR IFD SERIAL NO ......... 12639860
```

By clicking on the "Get Reader Info" button the web page will receive the info lines:

```
function CommandRInfo_onclick () {
  var bbb=EventsList.value;
  if (Check5.checked) {
    ss=SCardX_Easy.GetReaderInfoFmt(RList.value);
  }else{
    ss=SCardX_Easy.GetReaderInfo(RList.value);
  };
  var aaa = ss + strNewLine;
  EventsList.value= aaa + bbb;
  AddMessage("Get Reader Info", RList.value);
}
```

Receiving the Card Info

Add the "Get Card Info" and "Get Card ATR" button and the "Format Info" checkbox on the web page.

The SCardX Easy has two functions for the card info receiving:

- `GetCardInfo`
- `GetCardInfoFmt`

The function `GetCardInfo` returns the not formatted info lines like these ones:

```
[ICC STATE]
ATR STRING=3B 79 94 00 00 59 01 01 0F 01 00 01 04 A9
ICC PRESENCE=2
ICC INTERFACE STATUS=255
ICC TYPE PER ATR=1
CURRENT IO STATE=< no info >
[PROTOCOL]
DEFAULT DATA RATE=9624
MAX DATA RATE=115484
ASYNC PROTOCOL TYPES=3
DEFAULT CLK=3580
```

The function `GetCardInfoFmt` returns the formatted info lines like these ones:

```
ICC STATE
```
ATR STRING ..................... 3B 79 94 00 00 59 01 01 0F 01 00 01 04 A9
ICC PRESENCE .................... 2
ICC INTERFACE STATUS .......... 255
ICC TYPE PER ATR ............... 1
CURRENT IO STATE ............... < no info >

PROTOCOL
DEFAULT DATA RATE ............ 9624
MAX DATA RATE ................. 115484
ASYNC PROTOCOL TYPES ........ 3
DEFAULT CLK .................... 3580

By clicking on the "Get Card Info" button the web page will receive the info lines:

```javascript
function CommandCInfo_onclick() {
    var bbb=EventsList.value;
    if (Check6.checked) {
        ss=SCardX_Easy.GetCardInfoFmt(RList.value);
    }else{
        ss=SCardX_Easy.GetCardInfo(RList.value);
    }
    var aaa= ss + strNewLine;
    EventsList.value= aaa + bbb;
    AddMessage( "Get Card Info", RList.value);
}
```

By clicking on the "Get Card ATR" button the web page will receive the ATR string of the opened card:

```javascript
function CommandATR_onclick() {
    var ss=SCardX_Easy.GetCardATR(RList.value);
    AddMessage( "Get Card ATR", RList.value);
}
```

**Command APDU sending**

Add to the web page the following controls:

By clicking on the "Send" button the web page gets the hexadecimal parts of a command APDU according to ISO-7816 from the web page text input controls and puts its to parameters of the SCardX Easy function `SendCardAPDU`:

```javascript
function CommandAPDU_onclick() {
    var s = "";
    var ss = "";
    DataOut.value = "";
```
var sss=SCardX_Easy.SendCardAPDU( RList.value, Cla.value, Ins.value, P1_1.value, 
P2_1.value, P3_1.value, Le_1.value, DataIn.value,s,ss);

DataOut.value = sss;
}

This function returns the card's response APDU data buffer in the hexadecimal format according to ISO-7816.
The returned data placed in the text input control labeled “Received Data”.

**Unformatted data buffers sending**

Add to the web page the following controls:

```
Data buffer sending

Data buffer for sending
00 A4 00 00 02 3F00

Received Data
```

By clicking on the “Send” button the web page gets the hexadecimal value of the send buffer labeled as “Data buffer for sending” and puts it to a parameter of the SCardX Easy function SendCardDATA:

```
function CommandDATA_onclick() {
  ReceivedBuffer.value = "";
  var sss=SCardX_Easy.SendCardDATA( RList.value, SendBuffer.value);
  ReceivedBuffer.value = sss;
}
```

This function returns the card's answer on the sent data in the hexadecimal format. The returned data placed in the text input control labeled “Received Data”.

Copyright © 2005 by SCardSOFT
4.7 Tray Icon

The SCardX Easy ActiveX control allows you to manage the tray icon pop-up menu items and to receive the tray icon events.

Preparing the web page controls

Add to the web page the following controls:

Creating your own tray icon menu

The new pop-up menu of the SCardX Easy tray icon creates easy by calling of the TrayIconMenuCreate function.

You need to prepare the menu items list according to these rules:

- each new line in the list is the new menu item template;
- each menu item template consists of two parts;
  - the menu item ID;
  - the menu item caption;
- the parts of the menu item template are divided by the "=" character;
- if the menu item template begins with a "+" character the menus divider will be created;

Use the TryIconMenuItems text area control on the web page for preparing of the menu items list before menu creating.

For example your menu items list may be prepared like this:

ID_1=My Menu Item 1
- it's the divider
ID_2=My Menu Item 2
ID_3=My Menu Item 3

By clicking on the "Recreate TryIcon Menu" button the SCardX Easy will recreate its tray icon pop-up menu:

```javascript
function TryIconMenuCreate_onclick() {
    SCardX_Easy.TrayIconMenuCreate(TryIconMenuItems.value);
}
```
Changing the menu item properties

You can set up the following menu item properties:

- checked / unchecked
- enabled / disabled
- default / standard

All functions for changing of the menu item's properties needs the item ID string as a parameter.

Setting up the menu item as checked / unchecked

By clicking on the "Checked" button the SCardX Easy makes the menu item as checked:

```javascript
function TryIconMI_Check_onclick() {
    SCardX_Easy.TrayIconMenuItemSetChecked (ItemID.value, true);
}
```

By clicking on the "Unchecked" button the SCardX Easy makes the menu item as unchecked:

```javascript
function TryIconMI_Check_onclick() {
    SCardX_Easy.TrayIconMenuItemSetChecked (ItemID.value, false);
}
```

Setting up the menu item as enabled / disabled
By clicking on the "Disabled" button the SCardX Easy makes the menu item as disabled:

```javascript
function TryIconMI_Enabled_onclick() {
    SCardX_Easy.TrayIconMenuItemSetEnabled (ItemID.value, false);
}
```

By clicking on the "Enabled" button the SCardX Easy makes the menu item as enabled:

```javascript
function TryIconMI_Enabled_onclick() {
    SCardX_Easy.TrayIconMenuItemSetEnabled (ItemID.value, true);
}
```

**Setting up the menu item as default / standart**

By clicking on the "Default" button the SCardX Easy makes the menu item as default:

```javascript
function TryIconMI_Default_onclick() {
    SCardX_Easy.TrayIconMenuItemSetDefault (ItemID.value, true);
}
```

By clicking on the "Standart" button the SCardX Easy makes the menu item as standart menu item:

```javascript
function TryIconMI_Default_onclick() {
    SCardX_Easy.TrayIconMenuItemSetDefault (ItemID.value, false);
}
```

**Receiving the tray icon menu events**
The SCardX Easy creates the `OnTrayIconMenuItem` event when user clicks on the menu item:

```html
<SCRIPT LANGUAGE=javascript FOR=SCardX_Easy EVENT="OnTrayIconMenuItem( ItemID, IsChecked, IsEnabled, IsDefault, Caption )">
玺阝—
AddMessage("OnTrayIconMenuItem", ItemID );
//--->
</SCRIPT>
```

**Receiving the tray icon mouse double click event**

The SCardX Easy creates the `OnTrayIconDblClick` event when user double clicks on the tray icon:

```html
<SCRIPT LANGUAGE=javascript FOR=SCardX_Easy EVENT="OnTrayIconDblClick">
玺阝—
AddMessage("OnTrayIconDblClick", "");
//--->
</SCRIPT>
```
4.8 LookUp service

The SCardX Easy allows you to decode the system error codes and the reader states codes from its numerical values to its string descriptions.

Add to the web page the following controls:

![LookUp](image.png)

**Error LookUp**

By clicking on the "Error LookUp" button the web page calls the lookup function and receives the decoded value:

```javascript
function CommandErrorLookUp_onclick() {
    var bbb = EventsList.value;
    var ss = SCardX_Easy.LookUpError(TextELookUp.value);
    var aaa = ss + strNewLine;
    EventsList.value = aaa + bbb;
    AddMessage( "LookUpError", TextELookUp.value);
}
```

**Reader State LookUp**

By clicking on the "State LookUp" button the web page calls the lookup function and receives the decoded value:

```javascript
function CommandStateLookUp_onclick() {
    var bbb = EventsList.value;
    var ss = SCardX_Easy.LookUpReaderState(TextSLookUp.value);
    var aaa = ss + strNewLine;
    EventsList.value = aaa + bbb;
    AddMessage( "LookUpReaderState", "");
}
```

4.9 Data ciphering

The SCardX Easy ActiveX control allows you to encrypt and to decrypt the text strings using the DES algorithm.
Add to the web page the following controls:

![DES Data Cyphering](image)

Before using the ciphering functions you need to prepare the Key value in the hexadecimal format.

**Key rules:**

- if you want to use as a key ASCII symbols like the letters or numbers you need to convert its char codes to a hexadecimal format; for example the ASCII text "MyKey123" in the hex format is "4D794B6579313233"
- the length of the binary key always must be 8 bytes and the length of the key in the hexadecimal format always must be 16 hex symbols!

Create the new key and place its hex value into the text input control labeled "DES Key (Hex)".

**DES data encoding**

Type any text you like into the text control labeled "Text string for encrypt".

By clicking on the "Encrypt" button the web page takes the key hex value and the text for encrypt from the input controls and calls the `DES_EncryptString` encrypt function:

```javascript
function DESEnc_onclick() {
    var ss = SCardX_Easy.DES_EncryptString(KeyHEX.value, textDESTextForEnc.value);
    textDESTextAfterEnc.value = ss;
}
```

Encrypting example:

```
DES Key : AE9601A32FBCA85F
Text : Demo text for encrypt
Encrypted data : D6 D1 DB 24 59 SB 3A 9F 4D 22 58 9E 68 92 AB 29 40 41 16 B4 69 64 15 28
```

**DES data decoding**

Type the previous encrypted text as a hex buffer into the text control labeled "Encrypted data (Hex)".
By clicking on the "Decrypt" button the web page takes the key hex value and the encrypted hex buffer from the input controls and calls the `DES_DecryptString` decrypt function:

```javascript
function DESDec_onclick() {
    var ss = SCardX_Easy.DES_DecryptString(KeyHEX.value, textDESTextForDec.value);
    textDESTextAfterDec.value = ss;
}
```

Decrypting example:

<table>
<thead>
<tr>
<th>DES Key</th>
<th>8CA64DE9C1B23A7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decrypted text</td>
<td>Decrypt demo text</td>
</tr>
<tr>
<td>Encrypted data</td>
<td>BA 40 AC 81 34 9A DC AF 60 OB D5 EC 49 86 F8 90 7B B0 71 C1 05 38 A9</td>
</tr>
</tbody>
</table>

4.10 Configuring the web page startup

The web page startup is a good moment for setting up the SCardX Easy's properties.

```javascript
function window_onload() {
    // setting up the user interface properties
    SCardX_Easy.ActivePage = 0;
    // connecting the service
    SCardX_Easy.SmartCardService = 1;
    SCardX_Easy.ConnectionState = 1;
    // enabling/disabling the controls
    EnableControls();
}
```

We recommend you to set up the user interface properties of the SCardX Easy like the `BorderStyle` and other on the web page startup event.

4.11 Configuring the web page shutdown

Important!

You must call the `finalization function` of the SCardX Easy on the web page shutdown!

```javascript
function window_onunload() {
    SCardX_Easy.Finalize();
}
```
4.12 Tell: "Hello, cards World!"

Ok. Your first smart card web page is already prepared and ready to start!

ISO-7816 standard and smart card basics

The ISO-7816 is a base of the smart cards functionality. All another smart cards specifications were created under this standard and expands it only.

The card command may be sent into a card as a data buffer which is formatted as a command APDU (Application Protocol Data Unit).

The card's answer on each command APDU is the data buffer which is formatted as a response APDU.

According to ISO-7816-4 5.3.1 the command APDU consists of:
- a mandatory header of 4 bytes: Cla Ins P1 P2;
- a conditional body of a variable length;

Command APDU structure:

<table>
<thead>
<tr>
<th>Header</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cla Ins P1 P2</td>
<td>[Lc field] [DataIn field] [Le field]</td>
</tr>
</tbody>
</table>

What is the command APDU content?

According to ISO-7816-4 5.4 the command APDU contents:

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cla</td>
<td>Class</td>
<td>1</td>
<td>Class of instruction</td>
</tr>
<tr>
<td>Ins</td>
<td>Instruction</td>
<td>1</td>
<td>Instruction code</td>
</tr>
<tr>
<td>P1</td>
<td>Parameter1</td>
<td>1</td>
<td>Instruction parameter 1</td>
</tr>
<tr>
<td>P2</td>
<td>Parameter2</td>
<td>1</td>
<td>Instruction parameter 2</td>
</tr>
<tr>
<td>P3/Lc field</td>
<td>Length</td>
<td>variable</td>
<td>Number of bytes present in the data field of the command</td>
</tr>
<tr>
<td>DataIn field</td>
<td>Data</td>
<td>variable</td>
<td>String of bytes sent in the data field of the command</td>
</tr>
<tr>
<td>Le field</td>
<td>Length</td>
<td>variable &lt;= 3</td>
<td>Maximum number of bytes inspected in the data field of the response to the command</td>
</tr>
</tbody>
</table>

So each command is an APDU-formatted array of bytes which may be sent into a card.

What happens after the data was sent?

The card answers on the sent command APDU by its response APDU.

According to ISO-7816-4 5.3.3 the response APDU consists of:
• a conditional body of a variable length;
• a mandatory trailer of 4 bytes (status word) : SW1 SW2 ;

What is the response APDU content?

According to ISO-7816-4 5.4 the response APDU contents :

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataOut field</td>
<td>Data</td>
<td>variable = Le</td>
<td>String of bytes received in the data field of the response</td>
</tr>
<tr>
<td>SW1</td>
<td>Status byte 1</td>
<td>1</td>
<td>Command processing status</td>
</tr>
<tr>
<td>SW2</td>
<td>Status byte 2</td>
<td>1</td>
<td>Command processing qualifier</td>
</tr>
</tbody>
</table>

How it works?

For preparing of the command you need only to fill up the command APDU fields according to the card command which you need send into the card. Where can you find the values of these fields? You may find all necessary info about the command APDU and response APDU fields' values in the specifications of your smart cards.

The ISO-7816 standard defines the global principles of the card's functionality only.

The real cards always differs by its available commands’ set and by the values of the command APDU fields and all cards differs by its response APDU fields values.

But all chip smart cards always receives the commands as command APDU’s and answers back by the response APDU’s according to ISO-7816.

Please look more about the smart cards basics into the ISO-7816 standard and into the your cards’ specifications.

**Your first smart card command**

As example we’ll use the GSM SIM card and the GSM11.11 card specification.

According to ISO-7816 any smart card must have the Master File (MF) named 3F00. It's the "root directory" of the smart card's filesystem. The SIM card has the "3F00" file too.

We’ll try to send to the SIM card the command SELECT MF.

According to GSM11.11 9.2.1 the command APDU for the command SELECT is defined as:
9.2.1 SELECT

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>CLASS</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT</td>
<td>'A0'</td>
<td>'A4'</td>
<td>'00'</td>
<td>'00'</td>
<td>'02'</td>
</tr>
</tbody>
</table>

And according to GSM11.11 9.4.1 the successful respond APDU is defined as:

9.4 Status conditions returned by the card
This subclause specifies the coding of the status words SW1 and SW2.

9.4.1 Responses to commands which are correctly executed

<table>
<thead>
<tr>
<th>SW1</th>
<th>SW2</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
<td>'00'</td>
<td>normal ending of the command</td>
</tr>
<tr>
<td>9F</td>
<td>'XX'</td>
<td>normal ending of the command, with extra information from the proactive SIM containing a command for the ME. Length 'XX' of the response data</td>
</tr>
<tr>
<td>9F</td>
<td>'XX'</td>
<td>length 'XX' of the response data given in case of a SIM data download error</td>
</tr>
<tr>
<td>9F</td>
<td>'XX'</td>
<td>length 'XX' of the response data</td>
</tr>
</tbody>
</table>

So, according to GSM11.11 our command APDU is:

`Cla = A0`
`Ins = A4`
`P1 = 00`
`P2 = 00`
`P3/Lc = 02`
`DataIn = 3F00`

And after of this command APDU will be sent into the card you may receive from the card the following response APDU:

`DataOut = <none>`
`SW1 SW2 = 9F XX` (where XX is the length of the response data)

You can test this command using prepared smart card web page:

1. open the web page in the MS Internet Explorer;
2. connect to the service;
3. insert the card into a reader;
4. after the card will be opened by SCardX Easy please select your reader in the readers list on the web page;
5. fill up the fields of the "APDU Sending" controls on the web page according to the command APDU which was defined before;
6. click on the "Send APDU" button;
7. after the command sending please look on the text input control labeled as "SW1SW2" - there is the status word hex value like "9F17" must present there;

That's all.

1. you have prepared your first command APDU;
2. you have sent this command into the card;
3. and you have received from the card its answer on your command!

Congratulations!
At this moment you already have told to your SIM card - "Hello, cards World !"
5 SCardX Easy interface specification

5.1 Properties

User interface properties

- ActivePage
- BorderStyle
- BorderWidth
- EventsHistoryEnabled
- EventsLogging
- Visible
- VisibleEventsHistory
- VisibleStatusBar
- VisibleToolBar
- VisibleTrayIcon

Smart card work properties

- ConnectionState
- SmartCardService
- SeparateReceivedBytes

5.1.1 ActivePage

Specifies what the page of SCardX Easy is on the front of the control.
Description

Use the ActivePage property to determine what page is on the front of the control.

Type:

```
C++    : int
Basic  : As Long
Delphi : Integer
```

Possible values:

- `apStates` = $00000000
- `apEventsHistory` = $00000001

JavaScript syntax:

```
var apStates = 00000000;
var apEventsHistory = 00000001;

SCardX_Easy.ActivePage = apEventsHistory;
```
5.1.2 BorderStyle

Specifies the drawing style of the border of the SCardX Easy control.

![Image of border styles]

**Description**

Use the BorderStyle property for setting up the control's border style.

**Type:**

- C++ : int
- Basic : As Long
- Delphi : Integer

**Possible values:**

- afbNone = $00000000
- afbSingle = $00000001
- afbSunken = $00000002
- afbRaised = $00000003

**JavaScript syntax:**

```javascript
var afbNone = 00000000;
var afbSingle = 00000001;
var afbSunken = 00000002;
var afbRaised = 00000003;
SCardX_Easy.BorderStyle = afbSunken;
```

5.1.3 BorderWidth

Specifies the control's inner border width.
Description

Use the BorderWidth property for setting up the control's inner border width.

Type:

<table>
<thead>
<tr>
<th>Language</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++</td>
<td>int</td>
</tr>
<tr>
<td>Basic</td>
<td>As Long</td>
</tr>
<tr>
<td>Delphi</td>
<td>Integer</td>
</tr>
</tbody>
</table>

JavaScript syntax:

```javascript
SCardX_Easy.BorderWidth = 5;
```
5.1.4 **ConnectionState**

Specifies the current state of the connection to the selected smart card service.

**Description**

Use the ConnectionState property for connecting or disconnecting of the selected smart card service.

This property is unavailable while the SmartCardService is equal srv_Not_Defined.

**Type:**

<table>
<thead>
<tr>
<th>Language</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++</td>
<td>int</td>
</tr>
<tr>
<td>Basic</td>
<td>As Long</td>
</tr>
<tr>
<td>Delphi</td>
<td>Integer</td>
</tr>
</tbody>
</table>

**Possible values:**

- stServiceNotConnected = $00000000
- stServiceConnected = $00000001

**JavaScript syntax:**

```javascript
var stServiceNotConnected = 00000000;
var stServiceConnected = 00000001;
SCardX_Easy.ConnectionState = stServiceConnected;
```

5.1.5 **EventsHistoryEnabled**

Specifies whether the events history logging is enabled.

**Description**

Use the EventsHistoryEnabled property for enabling or disabling the logging of events on the Events History page.

**Type:**

<table>
<thead>
<tr>
<th>Language</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++</td>
<td>bool</td>
</tr>
<tr>
<td>Basic</td>
<td>As Boolean</td>
</tr>
<tr>
<td>Delphi</td>
<td>WordBool</td>
</tr>
</tbody>
</table>

**JavaScript syntax:**

```javascript
SCardX_Easy.EventsHistoryEnabled = true;
```
5.1.6  EventsLogging

Specifies the events logging mode.

Description

Use the EventsLogging property to determine the control's events logging mode.

Set the EventsLogging to xLog_AllEvents if you need more detailed events log.

Type:

```
C++    : int
Basic  : As Long
Delphi : Integer
```

Possible values:

- xLog_AllEvents = $00000000
- xLog_MostUsefulEvents = $00000001

JavaScript syntax:

```
var xLog_AllEvents         = 00000000;
var xLog_MostUsefulEvents  = 00000001;
SCardX_Easy.EventsLogging = xLog_MostUsefulEvents;
```

5.1.7  SeparateReceivedBytes

Specifies whether the received from the card hex bytes will be separated by the space character.

Description

Set the SeparateReceivedBytes property to true if you want to receive the separated bytes like this:

```
3B 79 94 00 00 59 01 01 0F 01
```

Otherwise the data will be received and showed like this:

```
3B799400005901010F01
```

Type:

```
C++    : bool
Basic  : As Boolean
Delphi : WordBool
```

JavaScript syntax:

```
SCardX_Easy.SeparateReceivedBytes = true;
```
5.1.8 **SmartCardService**

Specifies the selected smart card service.

**Description**

Use this property to change the selected smart card service.

If the srv_Not_DEFINED value assigned in this case the SCardX Easy closes all active connections and unloads the previous loaded service’s drivers.

If the srv_MS_PCSC_SCard_Service value assigned in this case the SCardX Easy tries to find the MS Smart Card service’s libraries and loads its.

After the service will be loaded you can connect of this service by assigning the value stServiceConnected to the ConnectionState property.

**Type:**

<table>
<thead>
<tr>
<th>C++</th>
<th>int</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>As Long</td>
</tr>
<tr>
<td>Delphi</td>
<td>Integer</td>
</tr>
</tbody>
</table>

**Possible values:**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>srv_Not_DEFINED</td>
<td>$00000000</td>
</tr>
<tr>
<td>srv_MS_PCSC_SCard_Service</td>
<td>$00000001</td>
</tr>
</tbody>
</table>

**JavaScript syntax:**

```javascript
var srv_Not_DEFINED = 00000000;
var srv_MS_PCSC_SCard_Service = 00000001;

SCardX_Easy.SmartCardService = srv_MS_PCSC_SCard_Service;
```
5.1.9 Visible

Specifies the SCardX Easy control’s visibility.

**Description**

Set the Visible property to false if you wish to hide the control on your web page.

**Type:**

- C++ : bool
- Basic : As Boolean
- Delphi : WordBool

**JavaScript syntax:**

```javascript
SCardX_Easy.Visible = true;
```

5.1.10 VisibleEventsHistory

Specifies the visibility of the “Events History” panel.

```
<table>
<thead>
<tr>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="..." alt="Connected" /></td>
<td><img src="..." alt="Connected" /></td>
</tr>
<tr>
<td><img src="..." alt="Events History" /></td>
<td><img src="..." alt="Events History" /></td>
</tr>
<tr>
<td><img src="..." alt="0x001F0122" /></td>
<td><img src="..." alt="0x001F0122" /></td>
</tr>
</tbody>
</table>
```

**Description**

Use the VisibleEventsHistory property for showing or hiding the “Events History” panel of the control.

**Type:**

- C++ : bool
- Basic : As Boolean
- Delphi : WordBool

**JavaScript syntax:**

```javascript
SCardX_Easy.VisibleEventsHistory = true;
```
5.1.11 VisibleStatusBar

Specifies the visibility of the status bar of the SCardX Easy.

Description

Use the VisibleStatusBar property for showing or hiding the status bar of the control.

Type:

- C++ : bool
- Basic : As Boolean
- Delphi : WordBool

JavaScript syntax:

```javascript
SCardX_Easy.VisibleStatusBar = true;
```

5.1.12 VisibleToolBar

Specifies the visibility of the tool bar of the SCardX Easy.

Description
Use the VisibleToolBar property for showing or hiding the tool bar of the control.

Type:

```
C++ : bool
Basic : As Boolean
Delphi : WordBool
```

JavaScript syntax:

```
SCardX_Easy.VisibleToolBar = true;
```

### 5.1.13 VisibleTrayIcon

Specifies the visibility of the tray icon of the SCardX Easy.

**True**

![Menu Items](image)

**Description**

Use the VisibleTrayIcon property for showing or hiding the tray icon of the control.

**Warning!** You can hide the TrayIcon under the Site or Developer's License only!

Type:

```
C++ : bool
Basic : As Boolean
Delphi : WordBool
```

JavaScript syntax:

```
SCardX_Easy.VisibleTrayIcon = true;
```

### 5.2 Functions

User interface functions

- [EventsHistoryClear](image)
Smart card work functions

GetCardATR
GetCardInfo
GetCardInfoFmt
GetReaderInfo
GetReaderInfoFmt
GetReadersList
IsCardReady
ReopenReader
SendCardAPDU
SendCardDATA

Other functions

DES_DecryptString
DES_EncryptString
Finalize
IsLocked
LookUpError
LookUpReaderState
Version
VersionMajor
VersionMinor

5.2.1 DES_DecryptString

Decrypts the encrypted by DES algorithm hexadecimal data buffer.

Arguments / parameters
### Argument Name

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| KeyHEX        | C++ : BSTR  
                Basic : As String  
                Delphi : WideString | the hex data buffer of the DES key value;  
                                                                   - the length of the binary key always must 8 bytes and  
                                                                   - the length of the key in the hexadecimal format always  
                                                                   must 16 hex symbols;  
                                                                   - do not use ASCII symbols for the key value: always  
                                                                   use the hexadecimal format only; |
| EncryptedDataHEX | C++ : BSTR  
                     Basic : As String  
                     Delphi : WideString | the hex data buffer of the previously encrypted by DES  
                                                                             text string; |

All arguments are passed by reference.

### Returns

The function returns the decrypted text string.

<table>
<thead>
<tr>
<th>Returning value data type</th>
</tr>
</thead>
</table>
| C++ : BSTR  
    Basic : As String  
    Delphi : WideString |

### Decrypting example:

**DES Key:** 8CA64DE9C1B123A7  
**Decrypted text:** Decrypt demo text  
**Encrypted data:** BA 40 AC 43 81 34 9A DC AF 60 0B D5 EC 49 86 F8 90 7B B0 71 C1 05 38 A9

**JavaScript syntax:**

```javascript
var Key = "8CA64DE9C1B123A7";
var Encrypted_String = "BA 40 AC 43 81 34 9A DC AF 60 0B D5 EC 49 86 F8 90 7B B0 71 C1 05 38 A9";
var Decrypted_String = SCardX_Easy.DES_DecryptString(Key, Encrypted_String);
```

### 5.2.2 DES_EncryptString

Encrypts ASCII symbols text string by the DES algorithm.

**Arguments / parameters**
<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyHEX</td>
<td>C++ : BSTR</td>
<td>the hex data buffer of the DES key value;</td>
</tr>
<tr>
<td></td>
<td>Basic : As String</td>
<td>• the length of the binary key always must 8 bytes and</td>
</tr>
<tr>
<td></td>
<td>Delphi : WideString</td>
<td>the length of the key in the hexadecimal format always</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• do not use ASCII symbols for the key value : always</td>
</tr>
<tr>
<td></td>
<td></td>
<td>use the hexadecimal format only;</td>
</tr>
<tr>
<td>CryptString</td>
<td>C++ : BSTR</td>
<td>any text string for encrypt;</td>
</tr>
<tr>
<td></td>
<td>Basic : As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi : WideString</td>
<td></td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

**Returns**

The function returns the hex data buffer of the encrypted string.

<table>
<thead>
<tr>
<th>Returning value data type</th>
<th>C++ : BSTR</th>
<th>Basic : As String</th>
<th>Delphi : WideString</th>
</tr>
</thead>
</table>

**Encrypting example:**

DES Key : AE9601A32FBCA85F
Text : Demo text for encrypt
Encrypted data : D6 D1 DB 24 59 BB 3A 9F 4D 22 58 96 68 92 AB 29 40 41 16 B4 69 64 15 28

**JavaScript syntax:**

```javascript
var Key = "AE9601A32FBCA85F";
var Text_String = "Demo text for encrypt";
var Encrypted_String = SCardX_Easy.DES_EncryptString(Key, Text_String);
```
5.2.3 **EventsHistoryClear**

Deletes all events messages from the grid of the "Events History" page of the control.

**Arguments / parameters**

<none>

**Returns**

<none>

**JavaScript syntax:**

```
SCardX_Easy.EventsHistoryClear();
```

5.2.4 **Finalize**

Closes all opened connections and frees all used memory.

**Arguments / parameters**

<none>

**Returns**

<none>

**Description**

Always call this function on the web page shutdown!

After calling of this function the SCardX Easy becomes unusable and ready for closing.

**JavaScript syntax:**

```
SCardX_Easy.Finalize();
```

5.2.5 **GetCardATR**

Returns the ATR string of the opened smart card.

**Arguments / parameters**

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReaderName</td>
<td>C++ : BSTR Basic : As String Delphi : WideString</td>
<td>smart card reader name;</td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

**Returns**
The function returns the ATR string in a hexadecimal format.

<table>
<thead>
<tr>
<th>Returning value data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++   : BSTR</td>
</tr>
<tr>
<td>Basic : As String</td>
</tr>
<tr>
<td>Delphi : WideString</td>
</tr>
</tbody>
</table>

JavaScript syntax:

```javascript
var ReaderName = "SCM Microsystems Inc. CHIPDRIVE Serial 0";
var ATR_String = SCardX_Easy.GetCardATR(ReaderName);
```
5.2.6 GetCardInfo

Returns an information about the opened smart card.

Arguments / parameters

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReaderName</td>
<td>C++: BSTR Basic: As String Delphi: WideString</td>
<td>smart card reader name;</td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

Returns

The function returns the info string list.

<table>
<thead>
<tr>
<th>Returning value data type</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C++: BSTR Basic: As String Delphi: WideString</td>
<td></td>
</tr>
</tbody>
</table>

Description

This function returns the list of the strings which are divided by the line breaks symbols $\#13\#10$.

Each info line is formatted as a standard INI file like of this example:

```
[ICC STATE]
ATR STRING=3B 79 94 00 00 59 01 01 0F 01 00
ICC PRESENCE=2
ICC INTERFACE STATUS=255
ICC TYPE PER ATR=1
```

JavaScript syntax:

```javascript
var ReaderName = "SCM Microsystems Inc. CHIPDRIVE Serial 0";
var Card_Info_Strings = SCardX_Easy.GetCardInfo(ReaderName);
```

5.2.7 GetCardInfoFmt

Returns a formatted information about the opened smart card.

Arguments / parameters
<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| ReaderName    | C++ : BSTR  
               Basic : As String  
               Delphi : WideString | smart card reader name; |

All arguments are passed by reference.

Returns

The function returns the info string list.

<table>
<thead>
<tr>
<th>Returning value data type</th>
</tr>
</thead>
</table>
| C++ : BSTR  
Basic : As String  
Delphi : WideString |

Description

This function returns the list of the strings which are divided by the line breaks symbols #13#10.

Each info line is formatted and already prepared for displaying like of this example:

```
ICC STATE
ATR STRING ................... 3B 79 94 00 00 59 01 01 0F 01 00 01 04 A9
ICC PRESENCE .................. 2
ICC INTERFACE STATUS .......... 255
ICC TYPE PER ATR .............. 1
CURRENT IO STATE ............. < no info >
```

JavaScript syntax:

```
var ReaderName = "SCM Microsystems Inc. CHIPDRIVE Serial 0";
var Card_Info_Strings = SCardX_Easy.GetCardInfoFmt(ReaderName);
```

5.2.8 GetEventsHistory

Returns the events history strings list from the "Events History" page.

Arguments / parameters

<none>

Returns

The function returns the events history string list.
Description

This function returns the list of the events messages from the "Events History" page which are divided by the line breaks symbols $\#13\#10$:

```
N | Source Event                | Value                    | Event Time          | Date       \\
1 | MS Smart Card service      | Driver loaded            | 00:02:18          | 01-XXX-05  \\
2 | MS Smart Card service      | Service connected        | 00:02:18          | 01-XXX-05  \\
3 | AKS ifdh 0 Reader state    | changed 0x00000012 : There is not card in the reader | 00:02:18 | 01-XXX-05  \\
4 | AKS ifdh 1 Reader state    | changed 0x000000012 : There is not card in the reader | 00:02:18 | 01-XXX-05  \\
5 | SCM Microsystems Inc. CHIPDRIVE Serial 0 | Reader state changed 0x001E0012 : There is not card in the reader | 00:02:18 | 01-XXX-05  \\
```

All fields in the each string are divided by the Tab character $\#9$.

This function may be useful for the errors localization during debugging of the remote application (web page).

JavaScript syntax:

```javascript
var Events_History_Strings = SCardX_Easy.GetEventsHistory();
```

5.2.9 GetReaderInfo

Returns the information about the reader.

Arguments / parameters

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReaderName</td>
<td>C++ : BSTR</td>
<td>smart card reader name;</td>
</tr>
<tr>
<td></td>
<td>Basic : As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi : WideString</td>
<td></td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

Returns

The function returns the info string list.
<table>
<thead>
<tr>
<th>Returning Value Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++ : BSTR</td>
</tr>
<tr>
<td>Basic : As String</td>
</tr>
<tr>
<td>Delphi : WideString</td>
</tr>
</tbody>
</table>

**Description**

This function returns the list of the strings which are divided by the line breaks symbols `#13#10`.

Each info line is formatted as a standart INI file like of this example:

```
[VENDOR INFO]
VENDOR NAME=SCM Microsystems Inc.
VENDOR IFD TYPE=CHIPDRIVE Serial
VENDOR IFD VERSION=< no info >
VENDOR IFD SERIAL NO=12639860
```

**JavaScript syntax:**

```javascript
var ReaderName = "SCM Microsystems Inc. CHIPDRIVE Serial 0";
var Reader_Info_Strings = SCardX_Easy.GetReaderInfo(ReaderName);
```
5.2.10 GetReaderInfoFmt

Returns the formatted information about the reader.

Arguments / parameters

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReaderName</td>
<td>C++: BSTR</td>
<td>smart card reader name;</td>
</tr>
<tr>
<td></td>
<td>Basic: As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: WideString</td>
<td></td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

Returns

The function returns the info string list.

<table>
<thead>
<tr>
<th>Returning value data type</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C++: BSTR</td>
<td></td>
</tr>
<tr>
<td>Basic: As String</td>
<td></td>
</tr>
<tr>
<td>Delphi: WideString</td>
<td></td>
</tr>
</tbody>
</table>

Description

This function returns the list of the strings which are divided by the line breaks symbols #13#10.

Each info line is formatted and already prepared for displaying like of this example:

```
VENDOR INFO
 VENDOR NAME .................. SCM Microsystems Inc.
 VENDOR IFD TYPE .............. CHIPDRIVE Serial
 VENDOR IFD VERSION .......... < no info >
 VENDOR IFD SERIAL NO ........ 12639860
```

JavaScript syntax:

```
var ReaderName = "SCM Microsystems Inc. CHIPDRIVE Serial 0";
var Reader_Info_Strings = SCardX_Easy.GetReaderInfoFmt(ReaderName);
```

5.2.11 Get Readers List

Returns the list of the smart card readers' names which are attached to your PC.

Arguments / parameters

<none>

Returns

The function returns the readers names string list.
### Description

This function returns the list of the readers names which are divided by the line breaks symbols 
#13#10:

AKS ifdh 0  
AKS ifdh 1  
SCM Microsystems Inc. CHIPDRIVE Serial 0

**JavaScript syntax:**

```javascript
var Readers_Names_Strings = SCardX_Easy.GetReadersList();
```

### 5.2.12 IsCardReady

Specifies whether the card in the reader is opened.

#### Arguments / parameters

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReaderName</td>
<td>C++: BSTR</td>
<td>smart card reader name;</td>
</tr>
<tr>
<td></td>
<td>Basic: As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: WideString</td>
<td></td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

#### Returns

The function returns true or false depends to whether the card in the reader is opened.

**JavaScript syntax:**

```javascript
var ReaderName = "SCM Microsystems Inc. CHIPDRIVE Serial 0";
var CardReady  = SCardX_Easy.IsCardReady(ReaderName);
```
5.2.13 IsLocked

Specifies whether the SCardX Easy is locked for smart card service commands.

Arguments / parameters
<none>

Returns

The function returns true or false depends to whether the SCardX Easy is locked.

<table>
<thead>
<tr>
<th>Returning value data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++ : bool</td>
</tr>
<tr>
<td>Basic : As Boolean</td>
</tr>
<tr>
<td>Delphi : WordBool</td>
</tr>
</tbody>
</table>

JavaScript syntax:

```javascript
var Locked = SCardX_Easy.IsLocked();
```

5.2.14 LookUpError

Decodes the error string message from its numerical code.

Arguments / parameters

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ErrorCodeHex</td>
<td>C++ : BSTR Basic : As String Delphi : WideString</td>
<td></td>
</tr>
<tr>
<td>(input)</td>
<td></td>
<td>the hexadecimal value of an integer error code;</td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

Returns

The function returns the decoded error string.

<table>
<thead>
<tr>
<th>Returning value data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++ : BSTR</td>
</tr>
<tr>
<td>Basic : As String</td>
</tr>
<tr>
<td>Delphi : WideString</td>
</tr>
</tbody>
</table>

Description

You may decode any error value which you need because this function uses the system function of the your PC operation system.
5.2.15 **LookUpReaderState**

Decodes the string value of the card reader state from its numerical code.

**Arguments / parameters**

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StateCodeHex</td>
<td>C++</td>
<td>the hexadecimal value of an integer state code;</td>
</tr>
<tr>
<td>(input)</td>
<td>Basic</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi</td>
<td></td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

**Returns**

The function returns the decoded reader state string list.

<table>
<thead>
<tr>
<th>Returning value data type</th>
<th>C++</th>
<th>Basic</th>
<th>Delphi</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BSTR</td>
<td>As String</td>
<td>WideString</td>
</tr>
</tbody>
</table>

**Description**

This function returns the list of the strings which are divided by the line breaks symbols `\#13\#10`. Each state line is formatted as a standard INI file like of this example:

- 0x00000020=There is a card in the reader
- 0x00000100=The card in the reader is in use by one or more other applications, but may be connected to in shared mode

**JavaScript syntax:**

```javascript
var ReaderState_Code_Hex = "00000122";
var ReaderState_StringList = SCardX_Easy.LookUpReaderState(ReaderState_Code_Hex);
```

5.2.16 **ReopenReader**

Reopens the reader.

**Arguments / parameters**
### 5.2.17 SendCardAPDU

Sends the command APDU into the opened smart card and returns the card's answer as a response APDU.

#### Arguments / parameters

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReaderName</td>
<td>C++ : BSTR</td>
<td>smart card reader name;</td>
</tr>
<tr>
<td></td>
<td>Basic : As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi : WideString</td>
<td></td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

JavaScript syntax:

```javascript
var ReaderName = "SCM Microsystems Inc. CHIPDRIVE Serial 0";
SCardX_Easy.ReopenReader(ReaderName);
```
<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReaderName</td>
<td>C++: BSTR</td>
<td>smart card reader name;</td>
</tr>
<tr>
<td></td>
<td>Basic: As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: WideString</td>
<td></td>
</tr>
<tr>
<td>Cla</td>
<td>C++: BSTR</td>
<td>the Class hex byte of the command APDU;</td>
</tr>
<tr>
<td></td>
<td>Basic: As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: WideString</td>
<td></td>
</tr>
<tr>
<td>Ins</td>
<td>C++: BSTR</td>
<td>the Instruction hex byte of the command APDU;</td>
</tr>
<tr>
<td></td>
<td>Basic: As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: WideString</td>
<td></td>
</tr>
<tr>
<td>P1</td>
<td>C++: BSTR</td>
<td>the Parameter 1 hex byte of the command APDU;</td>
</tr>
<tr>
<td></td>
<td>Basic: As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: WideString</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>C++: BSTR</td>
<td>the Parameter 2 hex byte of the command APDU;</td>
</tr>
<tr>
<td></td>
<td>Basic: As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: WideString</td>
<td></td>
</tr>
<tr>
<td>P3Lc</td>
<td>C++: BSTR</td>
<td>the Length hex byte of the command APDU;</td>
</tr>
<tr>
<td></td>
<td>Basic: As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: WideString</td>
<td></td>
</tr>
<tr>
<td>DataIn</td>
<td>C++: BSTR</td>
<td>the Data hex buffer of the command APDU;</td>
</tr>
<tr>
<td></td>
<td>Basic: As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: WideString</td>
<td></td>
</tr>
<tr>
<td>Le</td>
<td>C++: BSTR</td>
<td>the Length hex byte of the command APDU;</td>
</tr>
<tr>
<td></td>
<td>Basic: As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: WideString</td>
<td></td>
</tr>
<tr>
<td>SW1SW2</td>
<td>C++: BSTR</td>
<td>the Status Word (status hex bytes 1 and 2) of the response APDU;</td>
</tr>
<tr>
<td></td>
<td>Basic: As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: WideString</td>
<td></td>
</tr>
<tr>
<td>DataOut</td>
<td>C++: BSTR</td>
<td>the Data hex buffer of the response APDU;</td>
</tr>
<tr>
<td></td>
<td>Basic: As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: WideString</td>
<td></td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

Returns

The function returns the complete response APDU buffer in a hexadecimal format.

<table>
<thead>
<tr>
<th>Returning value data type</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C++: BSTR</td>
<td></td>
</tr>
<tr>
<td>Basic: As String</td>
<td></td>
</tr>
<tr>
<td>Delphi: WideString</td>
<td></td>
</tr>
</tbody>
</table>
Description

Use this function for sending the command APDU's into an opened smart card and for receiving of its response APDU's.

**JavaScript syntax:**

```javascript
var ReaderName = "SCM Microsystems Inc. CHIPDRIVE Serial 0";
var Cla_HEX    = "00";
var Ins_HEX    = "A4";
var P1_HEX     = "00";
var P2_HEX     = "00";
var P3Lc_HEX   = "02";
var Le_HEX     = "";
var DataIn_HEX = "3F00";
var DataOut_HEX = "";
var SW1SW2_HEX = "";

var ReceivedBuffer_HEX = SCardX_Easy.SendCardAPDU(ReaderName, Cla_HEX, Ins_HEX, P1_HEX, P2_HEX, P3Lc_HEX, Le_HEX, DataIn_HEX, SW1SW2_HEX, DataOut_HEX);
```
5.2.18 SendCardDATA

Sends an unformatted data buffer into the opened card and returns the unformatted card’s answer.

Arguments / parameters

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReaderName</td>
<td>C++: BSTR</td>
<td>smart card reader name;</td>
</tr>
<tr>
<td></td>
<td>Basic: As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: WideString</td>
<td></td>
</tr>
<tr>
<td>(input)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SentDataBuffer</td>
<td>C++: BSTR</td>
<td>an unformatted send data buffer in a hexadecimal format;</td>
</tr>
<tr>
<td></td>
<td>Basic: As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: WideString</td>
<td></td>
</tr>
<tr>
<td>(input)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

Returns

The function returns an unformatted buffer of the card response data in a hexadecimal format.

<table>
<thead>
<tr>
<th>Returning value data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++ : BSTR</td>
</tr>
<tr>
<td>Basic : As String</td>
</tr>
<tr>
<td>Delphi : WideString</td>
</tr>
</tbody>
</table>

Description

Use this function for sending an unformatted data into an opened smart card.

JavaScript syntax:

```javascript
var ReaderName = "SCM Microsystems Inc. CHIPDRIVE Serial 0";
var SendBuffer_HEX = "00 A4 00 00 02 3F00";
var ReceivedBuffer_HEX = SCardX_Easy.SendCardDATA(ReaderName,SendBuffer_HEX);
```

5.2.19 SetPref_PCSC_OnCardDetect

Sets up the card detecting defaults for using of the MS Smart Card service.

Arguments / parameters
### Argument Name | Data Type | Description
---|---|---
AutoOpenReader | C++ : bool, Basic : As Boolean, Delphi : WordBool | determines whether the card will be opened after detection;
PreferredProtocol | C++ : int, Basic : As Long, Delphi : Integer | determines the preferred protocol which will be used for the card opening;
PreferredSharingMode | C++ : int, Basic : As Long, Delphi : Integer | determines the reader sharing mode which will be used for the card opening;
CardClosingMode | C++ : int, Basic : As Long, Delphi : Integer | determines the card closing mode which will be used by the command ReopenReader;

All arguments are passed by reference.

**Returns**

<none>

**Description**

Use this command for setting up the card detecting defaults via control's interface.

These preferences' changes become visible on the "States" page after calling of this function immediately:

**Possible values:**

**PreferredProtocol**

- xProto_Autodetect = $00000000
- xProto_T0 = $00000001
- xProto_T1 = $00000002
- xProto_RAW = $00000003
- xProto_Undefined = $00000004
- xProto_Default = $00000005

**PreferredSharingMode**

- xSharing_ShareReader = $00000000
- xSharing_ExclusiveUse = $00000001
- xSharing_DirectReaderControl = $00000002

**CardClosingMode**
xClosing_LeaveCard = $00000000
xClosing_ResetCard = $00000001
xClosing_UnpowerCard = $00000002
xClosing_EjectCard = $00000003

JavaScript syntax:

//PreferredProtocol
var xProto_Autodetect = 00000000;
var xProto_T0 = 00000001;
var xProto_T1 = 00000002;
var xProto_RAW = 00000003;
var xProto_Undefined = 00000004;
var xProto_Default = 00000005;

//PreferredSharingMode
var xSharing_ShareReader = 00000000;
var xSharing_ExclusiveUse = 00000001;
var xSharing_DirectReaderControl = 00000002;

//CardClosingMode
var xClosing_LeaveCard = 00000000;
var xClosing_ResetCard = 00000001;
var xClosing_UnpowerCard = 00000002;
var xClosing_EjectCard = 00000003;
var AutoOpenReader = false;
var Proto = xProto_T0;
var Sharing = xSharing_DirectReaderControl;
var Closing = xClosing_EjectCard;

SCardX_Easy.SetPref_PCSC_OnCardDetect (AutoOpenReader, Proto, Sharing, Closing);

5.2.20 TrayIconMenuClear

Clears the SCardX Easy tray icon's pop-up menu.

Arguments / parameters

<none>

Returns

<none>

JavaScript syntax:

SCardX_Easy.TrayIconMenuClear();
5.2.21 TrayIconMenuCreate

Creates the new pop-up menu of the SCardX Easy's tray icon.

Arguments / parameters

<table>
<thead>
<tr>
<th>Argument</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MenuItemsList</td>
<td>C++: BSTR</td>
<td>the string list of the new menu items' templates;</td>
</tr>
<tr>
<td></td>
<td>Basic: As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: WideString</td>
<td></td>
</tr>
</tbody>
</table>

Arguments are passed by reference.

Returns

<none>

Description

Before calling of this function you need to prepare the menu items' list according to these rules:

- all strings in this list are divided by the line breaks symbols \#13\#10;
- each new line in the list is the new menu item template;
- each menu item template consists of two parts;
  - the menu item ID;
  - the menu item caption;
- these two parts of the menu item template are divided by the "=" character;
- if the menu item template begins with a "." character the menus divider will be created;

For example your menu items list may be prepared like this one:

```
ID_1=My Menu Item 1
----
ID_2=My Menu Item 2
ID_3=My Menu Item 3
```

These new menu items becomes visible into the tray icon's pop-up menu immediately after calling of this function:

![Menu items example](image)

JavaScript syntax:

```
var strNewLine = "\n";
var MenuItemsList = "ID_1=My Menu Item 1" + strNewLine + "ID_2=My Menu Item 2";

SCardX_Easy.TrayIconMenuCreate(MenuItemsList);
```
### 5.2.22 TrayIconMenuItemSetChecked

Makes the menu item of the tray icon’s pop-up menu as checked or unchecked.

#### Arguments / parameters

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItemID</td>
<td>C++: BSTR, Basic: As String, Delphi: WideString</td>
<td>the ID string of the menu item which was defined by the TrayIconMenuCreate function;</td>
</tr>
<tr>
<td></td>
<td>(input)</td>
<td></td>
</tr>
<tr>
<td>IsChecked</td>
<td>C++: bool, Basic: As Boolean, Delphi: WordBool</td>
<td>the checking flag;</td>
</tr>
<tr>
<td></td>
<td>(input)</td>
<td></td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

#### Returns

The function returns true if the menu item was found and the command was successful.

<table>
<thead>
<tr>
<th>Returning value data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++: bool, Basic: As Boolean, Delphi: WordBool</td>
</tr>
</tbody>
</table>

#### Description

Use this function for marking of the created menu items as checked or unchecked:

![Image of tray icon menu]

JavaScript syntax:

```javascript
var ItemID = "ID_1";
var YesNo = true;

var MenuItemWasFound = SCardX_Easy.TrayIconMenuItemSetChecked(ItemID, YesNo);
```
5.2.23 TrayIconMenuItemSetDefault

Makes the menu item of the tray icon's pop-up menu as default or standart.

**Arguments / parameters**

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItemID</td>
<td>C++: BSTR, Basic: As String, Delphi: WideString</td>
<td>the ID string of the menu item which was defined by the TrayIconMenuCreate function;</td>
</tr>
<tr>
<td>IsDefault</td>
<td>C++: bool, Basic: As Boolean, Delphi: WordBool</td>
<td>the default item flag;</td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

**Returns**

The function returns true if the menu item was found and the command was successful.

**JavaScript syntax:**

```javascript
var ItemID     = "ID_1";
var YesNo      = true;
var MenuItemWasFound = SCardX_Easy.TrayIconMenuItemSetDefault(ItemID, YesNo);
```

**Description**

Use this function for marking of the created menu items as default or standart:
5.2.24 TrayIconMenuItemSetEnabled

Makes the menu item of the tray icon’s pop-up menu as enabled or disabled.

Arguments / parameters

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItemID</td>
<td>C++: BSTR, Basic: As String, Delphi: WideString</td>
<td>the ID string of the menu item which was defined by the TrayIconMenuCreate function;</td>
</tr>
<tr>
<td>IsEnabled</td>
<td>C++: bool, Basic: As Boolean, Delphi: WordBool</td>
<td>the enabling flag;</td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

Returns

The function returns true if the menu item was found and the command was successful.

Description

Use this function for marking of the created menu items as enabled or disabled:

JavaScript syntax:

```javascript
var ItemID = "ID_1";
var YesNo = true;
var MenuItemWasFound = SCardX_Easy.TrayIconMenuItemSetEnabled(ItemID, YesNo);
```
5.2.25 Version

Returns the SCardX Easy version string.

Arguments / parameters

<none>

Returns

The function returns the full version string like: Version 1.3

<table>
<thead>
<tr>
<th>Returning value</th>
<th>data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++</td>
<td>BSTR</td>
</tr>
<tr>
<td>Basic</td>
<td>As String</td>
</tr>
<tr>
<td>Delphi</td>
<td>WideString</td>
</tr>
</tbody>
</table>

JavaScript syntax:

```javascript
var VersionString = SCardX_Easy.Version();
```

5.2.26 VersionMajor

Returns the major digit of the SCardX Easy ActiveX control version.

Arguments / parameters

<none>

Returns

The function returns the integer value of the major digit of the control's version.

<table>
<thead>
<tr>
<th>Returning value</th>
<th>data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++</td>
<td>int</td>
</tr>
<tr>
<td>Basic</td>
<td>As Long</td>
</tr>
<tr>
<td>Delphi</td>
<td>Integer</td>
</tr>
</tbody>
</table>

JavaScript syntax:

```javascript
var VersionMajor = SCardX_Easy.VersionMajor();
```
5.2.27 VersionMinor

Returns the minor digit of the SCardX Easy ActiveX control version.

**Arguments / parameters**

<none>

**Returns**

The function returns The integer value of the minor digit of the control's version.

<table>
<thead>
<tr>
<th>Returning value data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++   : int</td>
</tr>
<tr>
<td>Basic : As Long</td>
</tr>
<tr>
<td>Delphi : Integer</td>
</tr>
</tbody>
</table>

**JavaScript syntax:**

```javascript
var VersionMinor = SCardX_Easy.VersionMinor();
```
5.3 Events

User interface events

OnHistoryEvent
OnReaderSelected
OnTrayIconDbClick
OnTrayIconMenuItem

Smart card work events

OnCardDetected
OnCardInvalid
OnCardWait
OnConnected
OnDataSent
OnDisconnected
OnReadersList
OnReaderStateChanged

Other events

OnERROR
OnLock
OnUnlock
5.3.1 OnCardDetected

Occurs when the card was detected in the reader.

Arguments / parameters

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReaderName</td>
<td></td>
<td>smart card reader name;</td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

JavaScript syntax:

```html
<SCRIPT LANGUAGE=javascript FOR=SCardX_Easy EVENT="OnCardDetected(ReaderName)">
<!--
// ... Your code here ...
//-->
</SCRIPT>
```

5.3.2 OnCardInvalid

Occurs when the card was detected in the reader but the reader was not able to open it.

Arguments / parameters

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReaderName</td>
<td></td>
<td>smart card reader name;</td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

JavaScript syntax:

```html
<SCRIPT LANGUAGE=javascript FOR=SCardX_Easy EVENT="OnCardInvalid(ReaderName)">
<!--
// ... Your code here ...
//-->
</SCRIPT>
```
5.3.3 OnCardReady

Occurs when the card was detected and successfully opened in the reader.

Arguments / parameters

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReaderName</td>
<td>C++: BSTR Basic: As String Delphi: WideString</td>
<td>smart card reader name;</td>
</tr>
<tr>
<td></td>
<td>(output)</td>
<td></td>
</tr>
<tr>
<td>ATR</td>
<td>C++: BSTR Basic: As String Delphi: WideString</td>
<td>the ATR string of an opened card;</td>
</tr>
<tr>
<td></td>
<td>(output)</td>
<td></td>
</tr>
<tr>
<td>ProtocolValue</td>
<td>C++: int  Basic: As Long Delphi: Integer</td>
<td>the real active protocol code of an opened card;</td>
</tr>
<tr>
<td></td>
<td>(output)</td>
<td></td>
</tr>
<tr>
<td>Protocol</td>
<td>C++: BSTR Basic: As String Delphi: WideString</td>
<td>the real active protocol name of an opened card;</td>
</tr>
<tr>
<td></td>
<td>(output)</td>
<td></td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

JavaScript syntax:

```<SCRIPT LANGUAGE=javascript FOR=SCardX_Easy EVENT="OnCardReady (ReaderName, ATR, ProtocolValue, Protocol)">
<!-- ...
// ... Your code here ...
//-->
</SCRIPT>```
5.3.4 OnCardWait

Occurs when the card was removed from the reader.

Arguments / parameters

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReaderName</td>
<td>C++ : BSTR</td>
<td>smart card reader name;</td>
</tr>
<tr>
<td></td>
<td>Basic : As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi : WideString</td>
<td></td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

JavaScript syntax:

```javascript
<SCRIPT LANGUAGE=javascript FOR=SCardX_Easy EVENT="OnCardWait(ReaderName)">
<!--
// ... Your code here ...
//-->
</SCRIPT>
```
5.3.5 OnConnected

Occurs when the smart card service was successfully connected by SCardX Easy.

Arguments / parameters

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>C++ : int</td>
<td>the connected service code;</td>
</tr>
<tr>
<td></td>
<td>Basic : As Long</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi : Integer</td>
<td></td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

Possible values:
- srv_MS_PCSC_SCard_Service = $00000001

JavaScript syntax:

```javascript
<SCRIPT LANGUAGE=javascript FOR=SCardX_Easy EVENT="OnConnected( Service )">
<!--
// ... Your code here ...
//-->  
</SCRIPT>
```
5.3.6 OnDataSent

Occurs when the data was successfully sent into the opened smart card.

**Arguments / parameters**

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReaderName</td>
<td>C++: BSTR</td>
<td>smart card reader name;</td>
</tr>
<tr>
<td></td>
<td>Basic: As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: WideString</td>
<td></td>
</tr>
<tr>
<td>SentDataBuffer</td>
<td>C++: BSTR</td>
<td>an unformatted sent data buffer in a hexadecimal format;</td>
</tr>
<tr>
<td></td>
<td>Basic: As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: WideString</td>
<td></td>
</tr>
<tr>
<td>ReceivedDataBuffer</td>
<td>C++: BSTR</td>
<td>an unformatted received data buffer in a hexadecimal format;</td>
</tr>
<tr>
<td></td>
<td>Basic: As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: WideString</td>
<td></td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

**JavaScript syntax:**

```javascript
<SCRIPT LANGUAGE=javascript FOR=SCardX_Easy EVENT="OnDataSent(ReaderName,SentDataBuffer,ReceivedDataBuffer)">
<!--
// ... Your code here ...
//-->
</SCRIPT>
```

5.3.7 OnDisconnected

Occurs when the smart card service was disconnected.

**Arguments / parameters**

<none>

**JavaScript syntax:**

```javascript
<SCRIPT LANGUAGE=javascript FOR=SCardX_Easy EVENT=OnDisconnected>
<!--
// ... Your code here ...
//-->
</SCRIPT>
```
5.3.8 OnERROR

Occurs when the error was detected.

Arguments / parameters

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ErrorSource</td>
<td>C++ : BSTR Basic : As String Delphi : WideString</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( output )</td>
<td>the source where an error was detected by SCardX Easy;</td>
</tr>
<tr>
<td>ErrorCode</td>
<td>C++ : int Basic : As Long Delphi : Integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( output )</td>
<td>the integer error code value;</td>
</tr>
<tr>
<td>ErrorString</td>
<td>C++ : BSTR Basic : As String Delphi : WideString</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( output )</td>
<td>the decoded error string;</td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

JavaScript syntax:

```javascript
<SCRIPT LANGUAGE=javascript FOR=SCardX_Easy EVENT="OnERROR( ErrorSource , ErrorCode , ErrorString )">
// ... Your code here ...
</SCRIPT>
```

5.3.9 OnHistoryEvent

Occurs when the new event was added into the events grid of the "Events History" page.

Arguments / parameters
### Event ID

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventID</td>
<td>C++: <code>int</code></td>
<td>the number of the event line;</td>
</tr>
<tr>
<td></td>
<td>Basic: <code>As Long</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: <code>Integer</code></td>
<td></td>
</tr>
</tbody>
</table>

### Event Source

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventSource</td>
<td>C++: <code>BSTR</code></td>
<td>the source of the event;</td>
</tr>
<tr>
<td></td>
<td>Basic: <code>As String</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: <code>WideString</code></td>
<td></td>
</tr>
</tbody>
</table>

### Event Body

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventBody</td>
<td>C++: <code>BSTR</code></td>
<td>the event body message;</td>
</tr>
<tr>
<td></td>
<td>Basic: <code>As String</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: <code>WideString</code></td>
<td></td>
</tr>
</tbody>
</table>

### Event Value

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventValue</td>
<td>C++: <code>BSTR</code></td>
<td>the additional event info;</td>
</tr>
<tr>
<td></td>
<td>Basic: <code>As String</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: <code>WideString</code></td>
<td></td>
</tr>
</tbody>
</table>

### Event Time

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventTime</td>
<td>C++: <code>BSTR</code></td>
<td>the event time;</td>
</tr>
<tr>
<td></td>
<td>Basic: <code>As String</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: <code>WideString</code></td>
<td></td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

### Description

All parameters of this event are equal to the columns values of the events grid of the "Events History" page.

**JavaScript syntax:**

```javascript
<SCRIPT LANGUAGE=javascript FOR=SCardX_Easy EVENT="OnHistoryEvent(EventID, EventSource, EventBody, EventValue, EventTime)">
<!--
// ... Your code here ...
//-->
</SCRIPT>
```

### 5.3.10 OnLock

Occurs when the communication data exchange between the SCardX Easy and smart card service is active.

### Arguments / parameters

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message</td>
<td>C++: <code>BSTR</code></td>
<td>the string message about the current active operation;</td>
</tr>
<tr>
<td></td>
<td>Basic: <code>As String</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi: <code>WideString</code></td>
<td></td>
</tr>
</tbody>
</table>

All arguments are passed by reference.
JavaScript syntax:

```javascript
<SCRIPT LANGUAGE="javascript" FOR="SCardX_Easy" EVENT="OnLock(Message)">
<!--
// ... Your code here ...
//-->
</SCRIPT>
```

5.3.11 OnReaderSelected

Occurs when the user has selected the reader on the "States" page by mouse clicking on its item.

Arguments / parameters

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReaderName</td>
<td>C++ : BSTR</td>
<td>smart card reader name;</td>
</tr>
<tr>
<td></td>
<td>Basic : As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi : WideString</td>
<td></td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

JavaScript syntax:

```javascript
<SCRIPT LANGUAGE="javascript" FOR="SCardX_Easy" EVENT="OnReaderSelected(ReaderName)">
<!--
// ... Your code here ...
//-->
</SCRIPT>
```
5.3.12 OnReadersList

Occurs when the SCardX Easy receives the readers list from the smart card service.

Arguments / parameters

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadersList</td>
<td>C++ : BSTR</td>
<td>the list of the readers names which are divided by the line breaks symbols #13#10;</td>
</tr>
<tr>
<td></td>
<td>Basic : As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi : WideString</td>
<td></td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

JavaScript syntax:

```javascript
<SCRIPT LANGUAGE=javascript FOR=SCardX_Easy EVENT="OnReadersList(ReadersList)">
<!--
// ... Your code here ... 
//-->
</SCRIPT>
```

5.3.13 OnReaderStateChanged

Occurs when the reader state was changed.

Arguments / parameters

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReaderName</td>
<td>C++ : BSTR</td>
<td>smart card reader name;</td>
</tr>
<tr>
<td></td>
<td>Basic : As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi : WideString</td>
<td></td>
</tr>
<tr>
<td>ReaderState</td>
<td>C++ : BSTR</td>
<td>the new reader state integer code;</td>
</tr>
<tr>
<td></td>
<td>Basic : As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi : WideString</td>
<td></td>
</tr>
<tr>
<td>ReaderStateHex</td>
<td>C++ : int</td>
<td>the new reader state hex code;</td>
</tr>
<tr>
<td></td>
<td>Basic : As Long</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi : Integer</td>
<td></td>
</tr>
<tr>
<td>ReaderStateLookup</td>
<td>C++ : BSTR</td>
<td>the decoded new reader state string list; the strings are divided by the line breaks symbols #13#10;</td>
</tr>
<tr>
<td></td>
<td>Basic : As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delphi : WideString</td>
<td></td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

JavaScript syntax:

```javascript
<SCRIPT LANGUAGE=javascript FOR=SCardX_Easy EVENT="OnReaderStateChanged(ReaderName, ReaderState, ReaderStateHex, ReaderStateLookup)">
<!--
// ... Your code here ... 
//-->
</SCRIPT>
```
5.3.14 OnTrayIconDblClick

Occurs when the user double clicks on the tray icon of the SCardX Easy.

Arguments / parameters
<none>

JavaScript syntax:

```javascript
<SCRIPT LANGUAGE=javascript FOR=SCardX_Easy EVENT=OnTrayIconDblClick>
<!--
// ... Your code here ...
//-->
</SCRIPT>
```
5.3.15 OnTrayIconMenuItem

Occurs when the user clicks on the menu item of the tray icon's pop-up menu.

Arguments / parameters

<table>
<thead>
<tr>
<th>Argument Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItemID</td>
<td>C++: BSTR Basic: As String Delphi: WideString</td>
<td>the menu item ID string;</td>
</tr>
<tr>
<td></td>
<td>(output)</td>
<td></td>
</tr>
<tr>
<td>IsChecked</td>
<td>C++: bool Basic: As Boolean Delphi: WordBool</td>
<td>the item checked flag;</td>
</tr>
<tr>
<td></td>
<td>(output)</td>
<td></td>
</tr>
<tr>
<td>IsEnabled</td>
<td>C++: bool Basic: As Boolean Delphi: WordBool</td>
<td>the item enabled flag;</td>
</tr>
<tr>
<td></td>
<td>(output)</td>
<td></td>
</tr>
<tr>
<td>IsDefault</td>
<td>C++: bool Basic: As Boolean Delphi: WordBool</td>
<td>the item default flag;</td>
</tr>
<tr>
<td></td>
<td>(output)</td>
<td></td>
</tr>
<tr>
<td>Caption</td>
<td>C++: BSTR Basic: As String Delphi: WideString</td>
<td>the item caption;</td>
</tr>
<tr>
<td></td>
<td>(output)</td>
<td></td>
</tr>
</tbody>
</table>

All arguments are passed by reference.

JavaScript syntax:

```javascript
<SCRIPT LANGUAGE=javascript FOR=SCardX_Easy EVENT="OnTrayIconMenuItem( ItemID, IsChecked, IsEnabled, IsDefault, Caption )">
<!-- ...
// ... Your code here ...
//-->
</SCRIPT>
```
5.3.16 OnUnlock

Occurs when the communication data exchange between the SCardX Easy and smart card service was done and the control becomes ready for a new command.

**Arguments / parameters**

<none>

**JavaScript syntax:**

```javascript
<SCRIPT LANGUAGE=javascript FOR=SCardX_Easy EVENT=OnUnlock>
<!--
// ... Your code here ...
//-->
</SCRIPT>
```
6 Registration

6.1 Unregistered version limitations

Unregistered version of a SCardX Easy ActiveX control works as a demo version only.

These are the unregistered version limitations:

1. your program can send only from 7 up to 10 commands to a smart card per each SCardX Easy start;
2. the SCardX Easy shows unregistered version's reminders in the following areas:
   • in the License info item of the "States" page;
   • in the hint of the tray icon;
   • in the balloon of the tray icon;
3. you can't to hide the tray icon;
4. you may not contact the SCardX Easy support service;

6.2 Licensing

6.2.1 End-User Licenses

If you don't plan to re-distribute SCardX Easy ActiveX control in this case you may purchase one of our End-User Licenses:

1. **End-User Personal License** - personal usage by a single user;
2. **End-User Site License** - unlimited usage at a single company;

**Licences Prices**

[Purchase the Personal License](#)

[Purchase the Site License](#)

**End-User Personal License**

**Unlimited personal usage by a single user.**

You may create your own applications using SCardX Easy ActiveX control and to use its by yourself unlimited:

• license owner may create and unlimited use his own applications which are based on the SCardX Easy ActiveX control for his own personal tasks only;
• any re-distributions are not allowed;

**Registered Users Rights:**

After purchasing of the End-User Personal License you will be able:

• to unblock your copy of the SCardX Easy ActiveX control by your own Registration Certificate;
• to upgrade the new versions of the SCardX Easy ActiveX control for only 50% of the base price of the Personal License;
• to contact our support service for any questions about the SCardX Easy ActiveX control functionality or about the smart cards basics;
**End-User Site License**

**Unlimited usage at the single company**

By purchasing of this license you grants the SCardX Easy ActiveX control and all smart cards applications which are based on this ActiveX to all your developers and to all your company's staff at once.

For example SCardX Easy ActiveX control may be used by your corporate intranet smart cards oriented web site or by others your corporate smart cards applications:

- anybody may use the applications which are based on the SCardX Easy ActiveX control at the any of computers of a company which is an owner of this license;
- any re-distributions are not allowed;

**Registered Users Rights:**

After purchasing of the End-User Site License you will be able:

- to unblock your copy of the SCardX Easy ActiveX control by your own Registration Certificate;
- to upgrade the new versions of the SCardX Easy ActiveX control for only 50% of the base price of the Site License;
- to request the custom setup packs of the SCardX Easy ActiveX control like the web installation for free;
- to request the custom builds of the SCardX Easy ActiveX control according to your tasks; it may cost more depending on the requested functionality;
- to contact our support service for any questions about the SCardX Easy ActiveX control functionality or about the smart cards basics;

**6.2.2 Developers Licenses**

You may unlimited re-distribute SCardX Easy ActiveX control as a part of your own software solutions. In this case you may purchase one of our Developer's Licenses:

1. **Base Developer’s License** - unlimited re-distribution without source codes;
2. **Developer's License SC** - unlimited re-distribution with source codes included;
3. **Developer's License FULL** - unlimited re-distribution without copyright limitations;

Licences Prices

**Base Developers License**

**Unlimited re-distribution without source codes**

Any developer(s) may create applications using SCardX Easy ActiveX control and the licence owner may sale these applications unlimited without any additional payments to SCardSOFT:

- license owner may create, unlimited use and unlimited distribute the applications which are based on the SCardX Easy ActiveX control;
- re-distribution of SCardX Easy ActiveX control allowed as a part of license owner’s software without any additional payments to SCardSOFT;
- all rights on the SCardX Easy ActiveX control are reserved by its author;

**Developers License SC**

**Unlimited re-distribution with source codes included**
Any developer(s) may create applications using SCardX Easy ActiveX control and the licence owner may sale these applications unlimited without any additional payments to SCardSOFT:

- license owner may create, unlimited use and unlimited distribute the applications which are based on the SCardX Easy ActiveX control;
- re-distribution of SCardX Easy ActiveX control allowed as a part of license owner's software without any additional payments to SCardSOFT;
- all rights on the SCardX Easy ActiveX control are reserved by its author;
- full source codes of SCardX Easy ActiveX control are included;
- the copyright information of the SCardX Easy ActiveX control must be always included in the license of the software which uses the SCardX Easy ActiveX control;

**Developer License FULL**

**Unlimited re-distribution without copyright limitations**

Any developer(s) may create applications using SCardX Easy ActiveX control and the licence owner may sale these applications unlimited without any additional payments to SCardSOFT:

- license owner may create, unlimited use and unlimited distribute the applications which are based on the SCardX Easy ActiveX control;
- re-distribution of SCardX Easy ActiveX control allowed as a part of license owner's software without any additional payments to SCardSOFT;
- all rights on the SCardX Easy ActiveX control are reserved by its author;
- full source codes of SCardX Easy ActiveX control are included except of our shareware security subsystem;
- no copyright limitations are present; the control may be re-distributed without our copyright information visible;

**6.2.3 Custom versions**

**What software you can order?**

Additionally to our base solutions you can order the following custom software according to your specific tasks:
- custom versions of the SCardX Easy ActiveX control control;
- custom versions of the Smart Card ToolSet program;
- new smart card ActiveX controls;
- new smart card software;

**How much does it cost?**

The minimal fee for custom software order is a cost of the Site License. The real cost of your order will be calculated according to the requested functionality.

Please be ready to support us additionally, in the case if it will be necessary, by the following:
- a device(s) which will be used by an ordered software;
- smart cards which will be used by an ordered software;
- a device(s) and cards specification(s);

**Terms**

Our terms of a software creating are from two weeks up to some month depend on the requested functionality.
How to order?

Please read in details how to order a custom software versions on our web site.

6.3 Registration steps

6.3.1 Step 1: License Query

Run the program "SCardX Easy Control Center" from the start menu and make the following:

- open the "Registration" page;
- select "Step 1: I want now to create the License Query for receiving the Registration Certificate" and press on the "Go to Step 1: Create the License Query" button;
- fill up all information fields inside the "License Query Maker" window depending to the type of the License which you need and press on the "Make Query" button;
- Open the "License Query" page; there is the License Query's body text there;
- copy the License Query's text into a new e-mail letter and send it to SCardSOFT via e-mail: sales@scardsoft.com;

We will send you your own Registration Certificate after receiving of your money and after receiving of your License Query during a one working day.

6.3.2 Step 2: Purchasing the License

You can purchase the License on-line by your credit card.

Your payment will be processed by the Share-It! (Germany) internet payments’ service on the highest security level via a secure SSL connection.

Licences Prices
Purchase the License just now

Additionally we accepts the WebMoney and other transfers.

Read more how to purchase the License

We will send you your own Registration Certificate after receiving of your money and after receiving of your License Query during a one working day.

6.3.3 Step 3: Certificate registration

Copy the text of the Registration Certificate from the received our letter into a memory by "Copy" command.

Run the program "SCardX Easy Control Center" from the start menu and make the following:

- open the "Registration" page;
- select "Step 3: I already have my own Certificate and now I want to register the SCardX control" and press on the "Go to Step 3: Register the SCardX Easy control" button;
- paste the copied text of the received Registration Certificate into an opened "Certificate Registration Form" using the "Paste" button;
- register the program by pressing on the "Register SCardX Easy" button.
Index

- " " -
"Hello cards World !" 38

- A -
About 4
ActivePage property 42
Adding SCardX Easy to the new web page 22
APDU 38
ATR string receiving 55

- B -
BorderStyle property 44
BorderWidth property 44

- C -
Card detecting defaults setting up 68
Card Info example 27
Card Info receiving 57
Card Info receiving formatted 57
Card state checking 62
Clearing the Events History 55
Command APDU 38
Command APDU sending 65
Command APDU sending example 27
Connecting the service 46
Connection example 25
Connection testing 17
ConnectionState property 46
Contacts 4

- D -
Demo web page 21
DES decoding and encoding example 35
DES Decrypting 52
DES Encrypting 53
DES_DecryptString function 52
DES_EncryptString function 53
Disconnection example 25

- E -
Error message 83
Events History receiving 58
Events list 77
Events logging enabling/disabling 46
Events logging mode 47
Events receiving example 24
EventsHistoryClear function 55
EventsHistoryEnabled property 46
EventsLogging property 47
Examples path 21

- F -
Finalize example 37
Finalize function 55
First smart card web page : "Hello cards World !" 38
First smart card web page : Connection controls 25
First smart card web page : Data ciphering 35
First smart card web page : Events 24
First smart card web page : Interface functions 23
First smart card web page : LookUp 35
First smart card web page : New web page 22
First smart card web page : Opened reader controls 27
First smart card web page : Tray Icon 31
First smart card web page : Web page shutdown 37
First smart card web page : Web page startup 37
First start 17
Functions list 51

- G -
GetCardATR function 55
GetCardInfo function 57
GetCardInfoFmt function 57
GetEventsHistory function 58
GetReaderInfo function 59
GetReaderInfoFmt function 61
GetReadersList function 61
GSM11.11 38

- I -
IsCardReady function 62
IsLocked function 63
ISO-7816 38

- L -
Locked control checking 63
LookUp Error code 63
LookUp error code example 35
LookUp Reader State code 64
LookUp reader state code example 35
LookUpError function 63
LookUpReaderState function 64

- O -
OnCardDetected event 78
OnCardInvalid event 78
OnCardReady event 79
OnCardWait event 80
OnConnected event 81
OnDataSent event 82
OnDisconnected event 82
OnERROR event 83
OnHistoryEvent event 83
OnLock event 84
OnReaderSelected event 85
OnReadersList event 86
OnReaderStateChanged event 86
OnTrayIconDblClick event 87
OnTrayIconMenuItem event 88
OnUnlock event 89

- P -
Properties list 42

- R -
Reader Info example 27
Reader Info receiving 59
Reader Info receiving formatted 61
Readers list receiving 61
Readers list receiving example 25
Registration : Developers Licenses 91
Registration : End-User Licenses 90
Registration of the ActiveX control on the web page 14
Registration Step 1 : License Query 93
Registration Step 2 : Purchasing the License 93
Registration Step 3 : Certificate registration 93
Reopen Reader 64
Reopen reader example 27
ReopenReader function 64
Response APDU 38

- S -
SendCardAPDU function 65
SendCardDATA function 68
SeparateReceivedBytes property 47
Separating the received HEX bytes 47
SetPref_PSCS_OnCardDetect function 68
Show / hide the EventsHistory 49
Show / hide the StatusBar 50
Show / hide the ToolBar 50
Show / hide the Tray Icon 51
Smart card service selecting 48
SmartCardService property 48
Status word 38
SW1SW2 38

- T -
Tray Icon double click event 87
Tray Icon example 31
Tray Icon Menu : Clearing 70
Tray Icon Menu : Creating new 71
Tray Icon Menu Item : checked / unchecked 72
Tray Icon Menu Item : default / standart 73
Tray Icon Menu Item : enabled / disabled 74
Tray Icon Menu Item : events receiving 88
TrayIconMenuClear function 70
TrayIconMenuCreate function 71
TrayIconMenuItemSetChecked function 72
TrayIconMenuItemSetDefault function 73
TrayIconMenuItemSetEnabled function 74

- U -
Unformatted data buffer sending 68
Unformatted data buffers sending example 27
Unregistered version limitations 90

- V -
Version function 75
Version Major digit receiving 75
Version Minor digit receiving 76
Version string receiving 75
VersionMajor function 75
VersionMinor function 76
Visible property 49
VisibleEventsHistory property 49
VisibleStatusBar property 50
VisibleToolBar property 50
VisibleTrayIcon property 51

- W -

Web page shutdown example 37
Web page startup example 37