SAEAUT SMS Library (MFC)

Flexible DLL library that allow to send and receive SMS messages.

---

by SAE-Automation, s.r.o.

SAEAUT SMS Library

Production

Table of Contents

Foreword 0

Part I SAEAUT SMS Library (MFC) 4

1 Introduction 4
2 Main Features 5

Part II CGSMModem class 8

1 _DoReadSMS 11
2 _DoSendSMS 11
3 GSMModem 13
4 Connect 13
5 Disconnect 15
6 DoAT 16
7 DoATCommand 16
8 DoDeleteSMS 17
9 DoEnterPIN 18
10 DoEnterPIN2 18
11 DoEnterPUK 19
12 DoEnterPUK2 19
13 DoEnterServiceCenterAddress 20
14 DoGetSMSIndex 20
15 DoReadModemAnswer 21
ModemAnswerEnum 23
16 DoReadSMS 24
17 DoSendSMS 24
18 DoSetDefaultConfigurationZ 25
19 DoSetEcho 25
20 DoSetReportMobileEquipementErrors 26
21 GetComPortBaudRate 27
22 GetComPortDataBits 27
23 GetComPortName 27
24 GetComPortParity 28
25 GetDeliveryStatusText 28
26 GetErrorMessage 30
27 GetLastError 34
Error codes 38
28 GetModemManufacturer 42
29 GetModemModel 42

30 GetModemReadBuffer 43
31 GetRecvSMSDeliveryRefNumber 43
32 GetRecvSMSDeliveryStatus 44
Delivery status codes 45
33 GetRecvSMSSenderPhoneNumber 45
34 GetRecvSMSServiceCentreTimestamp 46
35 GetRecvSMSText 46
36 GetRecvSMSTimestamp 47
37 SetComPort 47
38 SetPIN 49
39 SetReadTimeout 49
40 SetServiceCenterAddress 50

Part III Sample Applications 52

1 Sample1 52

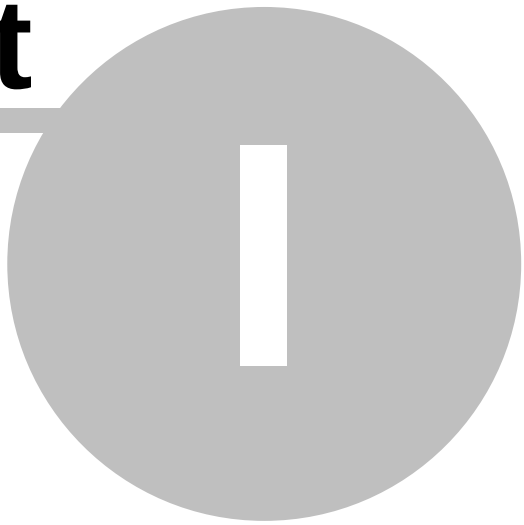
Part IV Known issues 57

Part V Support & Contact 59

Index 60

SAEAUT SNMP OPC Server

Part



1 SAEAUT SMS Library (MFC)

http://www.saeautom.sk
sae-automation@saeautom.sk

SAEAUT SMS Library (MFC)

Flexible DLL library that allow to send and receive SMS messages.

The library allows to add a new valuable functionality, processing SMS messages, to your own applications. All functionality which relates with sending, receiving SMS messages and configuring required parameters is enclosed in one CGSMModem class. The CGSMModem class is very good documented class based on Microsoft Foundation Class (MFC) Library. The SAEAUT SMS Library (MFC) is completed with practical samples.

We believe that using this DLL library you will be able to send and receive SMS messages from you applications during few minutes.

Related aticles on the web
SAEAUT SMS Service on the web
SAEAUT SMS Server on the web

Our other products on web
SAEAUT SNMP OPC Server on the web
OpcDbGateway on the web

See Also
Introduction, Main Features, CGSMModem class, Sample Applications

1.1 Introduction

This manual contains the information you need to get started with SAEAUT SMS Library (MFC). The exposed CGSMModem class allows you to easily perform sending and receiving of SMS messages through exposed methodes directly from your programs developed with **Microsoft Visual C++**.

This manual contains descriptions of sample application, which can be modified for your own applications.

Note that: MFC extension DLL

The **SMSLibMFC.DLL** library is created as **MFC extension DLL**. It means that only aplications developed with MFC (Microsoft Foundation Class library) can call functions from this **SMSLibMFC.DLL**.

To use this **SMSLibMFC.DLL**, you should already be familiar with **Windows Operating System** and programming environment **Microsoft Visual Studio.NET 2005** and **Microsoft Foundation Class (MFC)** library.

See Also

CGSMModem class, SAEAUT SMS Library (MFC)

1.2 Main Features

Communication

- Communication is established through **GSM modem**. **GSM** communication is the most popular standard for mobile phones in the world.
- SAEAUT SMS Library (MFC) wrappers the base **GSM modem** functionality to CGSMModem class. Through this class interface is possible:

  - to set all necessary communication parameters: serial port used, transfer speed, timeouts, etc.
  - to set all GSM parameters: PIN code and Service Center Address.
  - to send **SMS** message to a specific phone number,
  - to receive **SMS** message,
  - to send **AT commands** to GSM modem,
  - to receive a modem anwer from GSM modem.

- **GSM modem** can be connected to any personal computer via standard communication interfaces as follows:

  - RS-232,
  - IrDA,
  - Bluetooth,
  - USB (virtual serial port)

- Connection to **GSM network** is interposed by a mobile network operator's **SIM** card which is inserted in GSM modem device.

Sending SMS messages

- The maximal length of single SMS message is **160 characters**.
- SMS message can be sent to one or several phone numbers.
- Sent SMS is associated with reference number.

Reading received SMS messages

- The SAEAUT SMS Library (MFC) allows to read the received SMS of two following types:

  1. **Delivery report** about sent SMS message.
  2. **SMS messages which were sent by other user**.

Sending AT commands to GSM modem and receiving answer

- The library allows to send a specific AT commands to GSM modem and receive the answer from GSM modem.

Tests && Optimalizations

- The SAEAUT SMS Library (MFC) has been tested and optimized for services of following operators:

  - ORANGE,
  - T-MOBILE,
  - O2,
  - Of course, the system is open for use in other operators networks too.

- The SAEAUT SMS Library (MFC) has been tested and optimized for the GSM modems:

  - WAVECOM,
  - SAMBA,
  - FALCOM,
  - Of course, the system is open for using GSM modems from other vendors too.

See Also

CGSMModem class, SAEAUT SMS Library (MFC)

SAEAUT SNMP OPC Server

Part II

2 CGSMModem class

The class manages a connection to GSM modem and allows direct sending and receiving SMS messages.

```
class AFX_EXT_CLASS CGSMModem
```

Remarks

The **Connect** function is used to open COM port and check wheather modem is available.

Members

Construction

GSMModem	Public	The function is called when a CGSMModem object is constructed.
~CGSMModem	Public	The destructor.

Operations

_DoReadSMS	Protected	The function reads a specific SMS message.
_DoSendSMS	Protected	The function sends the SMS text to specified phone number.
Connect	Public	The function connects to a target modem.
Disconnect	Public	The function closes the COM port connection.
DoAT	Protected	The function sends the **AT+<CR><LF>** command to the COM port and wait for a modem answer.
DoATCommand	Public	The function send a specified AT command to COM port.
DoDeleteSMS	Protected	The function deletes a stored SMS message.
DoEnterPIN	Public	The function enters a specified PIN code.
DoEnterPIN2	Public	The function enters a specified PIN2 code.
DoEnterPUK	Public	The function enters a specified PUK code.
DoEnterPUK2	Public	The function enters a specified PUK2 code.
DoEnterServiceCenterAddress	Public	The function enters a specified service center address to modem.
DoGetSMSIndex	Protected	The function retrieves an index which points to the first available stored SMS message.
DoReadModemAnswer	Public	The function reads an modem answer from the COM port.
DoReadSMS	Public	The function reads the first available stored SMS message.
DoSendSMS	Public	The function sends the SMS text to specified phone number.
DoSetDefaultConfigurationZ	Protected	The function restores the configuration profile.
DoSetEcho	Protected	The function activates or deactivate the sending ECHO characters in modem answer. The function sends the specific AT command to the COM port and wait for a modem answer.
DoSetReportMobileEquipementErrors	Protected	The function enables or disables the use of result code.
GetComPortBaudRate	Public	The function retrieves the COM port baudrate parameter.
GetComPortDataBits	Public	The function retrieves the COM port databits parameter.
GetComPortName	Public	The function retrieves the COM port name.
GetComPortParity	Public	The function retrieves the COM port parity parameter.
GetDeliveryStatusText	Public	The function retrieves the SMS delivery status text.
GetErrorMessage	Public	The function retrieves the error message string for an input error code.
GetLastError	Public	The function retrieves the error code for

Operations (private)

SetLastError	Private	Internal: Sets an error code for the last operation that failed.
DoSetPreferredMessageFormat	Private	Internal: Sets an prefereed SMS message format.
DoInitModem	Private	Internal: Executes a specific modem initialization. The function calls three following functions in the order: 1. DoSetDefaultConfigurationZ, 2. DoSetEcho, 3. DoSetReportMobileEquipementErrors.
DoGetNetworkOperator	Private	Internal: Retrieves the modem read data in the special string format.
DoReadModemAnswerData	Private	Internal: Retrieves the modem read data in the special string format.

Data members

m_pComPort	Private	Contains a pointer to COM port object.
m_pLastError	Private	Contains an error code for the last operation that failed.
m_dwReadTimeout	Private	Contains an read timeout in mili-seconds.
m_strPIN	Private	Contains an PIN code.
m_strServiceCenterAddress	Private	Contains an service center number.
m_strRecvSMSTimestamp	Private	Contains the timestamp for the last received SMS message.
m_strRecvSMSServiceCentreTimestamp	Private	Contains the service centre timestamp for the last received SMS message.
m_strRecvSMSSenderPhoneNumber	Private	Contains the sender phone number for the last received SMS message.
m_strRecvSMSText	Private	Contains the SMS text for the last received SMS message.
m_nRecvSMSDeliveryStatus	Private	Contains the delivery status code for the last received SMS message.
m_nRecvSMSDeliveryRefNumber	Private	Contains the SMS reference number for the last received SMS message.
m_bLastDoSendOrDoReadOK	Private	Contains the flag whether the last SMS was sent or read successfully.

Enums

ModemAnswerEnum	The enem is available in header ModemAnswer.h.

Special codes

Error codes	The error codes.
Delivery status codes	The delivery status codes.

Example code

The Sample1 example shows the use of the **CGSMModem** class.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class,

2.1 _DoReadSMS

The function reads a specific SMS message.

```
int _DoReadSMS(
  int nSMSIndex = 1
 );
```

Parameters
nSMSIndex
[in] An index which points to the stored SMS message.

Return Values
The following return values are possible for **_DoReadSMS** function.

Value	Description
SMS_ERROR	Error occured. If the a value of SMS_ERROR is returned then a specific error code can be retrieved by calling GetLastError.
SMS_RECV_NOSMS	No SMS available for reading.
SMS_RECV_SUBMIT	Ok. A received SMS was read successful.
SMS_RECV_DELIVERY	Ok. A delivery SMS was read successful.

Remarks
The **_DoReadSMS** function is used to read stored SMS messages. Each SMS that was read sucessfull is immediately deleted.

Requirements
Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class, DoReadSMS

2.2 _DoSendSMS

The function sends the SMS text to specified phone number.

```
int DoSendSMS(
```

```
    LPCTSTR lpszPhoneNumber,
    LPCTSTR lpszSMSText,
  );
```

Parameters

lpszPhoneNumber

[in] Pointer to a null-terminated string that specifies the target phone number.

lpszSMSText

[in] Pointer to a null-terminated string that specifies the SMS text to be sent. The maximal SMS text lenght is 160 charactes.

Return Values

If no error occurs, **_DoSendSMS** returns the reference number of sent SMS. Otherwise, a value of SMS_ERROR is returned, and a specific error code can be retrieved by calling GetLastError.

Remarks

The **_DoSendSMS** function is used to send the SMS text to specified phone number.

The **SAEAUT SMS Library (MFC) DEMO** version adds to the SMS message prefix www.saeautom.eu/SMSLibrary/ and truncates a new SMS message on valid length (160 characters).

Example code

The following code example shows the use of the **_DoSendSMS** function.

The function sends the SMS text to specified phone number.

```
int DoSendSMS(LPCTSTR lpszPhoneNumber, LPCTSTR lpszSMSText, BYTE
nNumberOfRetries/*=3*/)
{
     // @flow0 | Retry max. three times to send SMS message
     for (BYTE nRetry = 0; nRetry < nNumberOfRetries; nRetry++)
     {
           // @flow0 | the last SMS was sent or read successfully.
           if( !m_bLastDoSendOrDoReadOK )
           {
                 // @flow0 | init modem
                 if( DoInitModem() == false )
                 {
                       continue;
                 }

                 // @flow0 | enter PIN code
                 if( DoEnterPIN(m_strPIN) == false )
                 {
                       return SMS_ERROR;
                 }

                 // @flow0 | setting message format
                 if( DoSetPreferredMessageFormat( 0 ) == false )
                 {
                       continue;
                 }

                 // @flow0 | enter the service center address
                 if(
DoEnterServiceCenterAddress(m_strServiceCenterAddress) == false )
                 {
```

```
                        continue;
                }
        }

        // @flow1 | Send the SMS message
        int nRefNumber = _DoSendSMS( lpszPhoneNumber, lpszSMSText );

        if ( nRefNumber != SMS_ERROR )
        {
                m_bLastDoSendOrDoReadOK = true;

                return nRefNumber;                  // Ok. SMS messages
was sent successfully.
        }

        m_bLastDoSendOrDoReadOK = false;          // the last SMS
sending failed
    }

    // @flow0 | FAULT !!! No SMS was sent successfully
    return SMS_ERROR;
}
```

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also

CGSMModem class, DoSendSMS

2.3 GSMModem

The function is called when a CGSMModem object is constructed.

```
CGSMModem( );
```

Example code

The Sample1 example shows the use of the **GSMModem** function.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also

CGSMModem class

2.4 Connect

The function connects to a target modem.

```
bool Connect( );

bool Connect(
  LPCTSTR lpszPortName,
  int nBaudRate = CBR_9600,
  int nDataBits = 8,
  int nStopBits = ONESTOPBIT,
  int nParity = NOPARITY
);
```

Parameters

lpszPortName

[in] Pointer to a null-terminated string that specifies the name of COM port.

nBaudRate

[in] Specifies the baud rate at which the communication device operates. It is an actual baud rate value, or one of the following baud rate indexes:

- CBR_110
- CBR_300
- CBR_600
- CBR_1200
- CBR_2400
- CBR_4800
- CBR_9600
- CBR_14400
- CBR_19200
- CBR_38400
- CBR_56000
- CBR_57600
- CBR_115200
- CBR_128000
- CBR_256000

nDataBits

[in] Specifies the number of bits in the bytes transmitted and received.

nStopBits

[in] Specifies the number of stop bits to be used. The following values are possible for this member.

Value	Description
ONESTOPBIT	1 stop bit
ONE5STOPBITS	1.5 stop bits
TWOSTOPBITS	2 stop bits

nParity

[in] Specifies the parity scheme to be used. The following values are possible for this member.

Value	Description
EVENPARITY	Even
MARKPARITY	Mark
NOPARITY	No parity
ODDPARITY	Odd
SPACEPARITY	Space

Return Values
Nonzero if the **Connect** function was successful. Otherwise, a value of **false** is returned, and a specific error code can be retrieved by calling GetLastError.

Remarks
The **Connect** function is used to open COM port and check wheather modem is available.

Example code
The Sample1 example shows the use of the **Connect** function.

Requirements
Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class, SetComPort

2.5 Disconnect

The function closes the COM port connection.

```
bool Disconnect( );
```

Return Values
Nonzero if the closing of COM port was successful. Otherwise, a value of **false** is returned.

Remarks
The **Disconnect** function is used to close COM port co.

Example code
The Sample1 example shows the use of the **Disconnect** function.

Requirements
Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class

2.6 DoAT

The function sends the **AT+<CR><LF>** command to the COM port and wait for a modem answer.

```
bool DoAT( );
```

Return Values

Nonzero if the **DoAT** function was successful. Otherwise, a value of **false** is returned, and a specific error code can be retrieved by calling GetLastError.

Remarks

The **DoAT** function is used to send a **AT+<CR><LF>** command to COM port and wait for a modem answer. The modem answer is evaluated via DoReadModemAnswer function.

Example code

The following code example shows the use of the **DoAT** function.

The function sends the **AT+<CR><LF>** command to the COM port and wait for a modem answer.

```
bool DoAT(void)
{
     // @flow0 | Send the AT command
     CString strATCommand = _T("AT\r");
     if ( DoATCommand(strATCommand) == false )
     {
          return false;
     }

     // @flow0 | Read an answer from modem
     if ( DoReadModemAnswer() == emaOKCRLF )
     {
          return true;
     }

     return false;
}
```

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also

CGSMModem class

2.7 DoATCommand

The function send a specified AT command to COM port.

```
bool DoATCommand(
  LPCTSTR lpszATCommand
);
```

Parameters
lpszATCommand
[in] Pointer to a null-terminated string that specifies the AT command.

Return Values
Nonzero if the **DoATCommand** function was successful. Otherwise, a value of **false** is returned, and a specific error code can be retrieved by calling GetLastError.

Remarks
The **DoATCommand** function is used to send a **AT** command to COM port.

Example code
The following code example shows the use of the **DoATCommand** function.

The function sends the **AT+<CR><LF>** command to the COM port and wait for a modem answer.

```
bool DoAT(void)
{
     // @flow0 | Send the AT command
     CString strATCommand = _T("AT\r");
     if ( DoATCommand(strATCommand) == false )
     {
          return false;
     }

     // @flow0 | Read an answer from modem
     if ( DoReadModemAnswer() == emaOKCRLF )
     {
          return true;
     }

     return false;
}
```

Requirements
Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class

2.8 DoDeleteSMS

The function deletes a stored SMS message.

```
bool DoDeleteSMS(
  int nSMSIndex
);
```

Parameters
nSMSIndex
[in] An index which points to the stored SMS message.

Return Values
Nonzero if the **DoDeleteSMS** function was successful. Otherwise, a value of **false** is returned, and a specific error code can be retrieved by calling GetLastError.

Remarks
The **DoDeleteSMS** function is used to delete a stored SMS message.

Requirements
Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class

2.9 DoEnterPIN

The function enters a specified PIN code.

```
bool DoEnterPIN(
  LPCTSTR lpszPIN
);
```

Parameters
lpszPIN
[in] Pointer to a null-terminated string that specifies the PIN code.

Return Values
Nonzero if the **DoEnterPIN** function was successful. Otherwise, a value of **false** is returned, and a specific error code can be retrieved by calling GetLastError.

Remarks
The **DoEnterPIN** function is used to enter a PIN code.

Requirements
Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class, SetPIN

2.10 DoEnterPIN2

The function enters a specified PIN2 code.

```
bool DoEnterPIN2(
  LPCTSTR lpszPIN2
);
```

Parameters

lpszPIN2
[in] Pointer to a null-terminated string that specifies the PIN2 code.

Return Values
Nonzero if the **DoEnterPIN2** function was successful. Otherwise, a value of **false** is returned, and a specific error code can be retrieved by calling GetLastError.

Remarks
The **DoEnterPIN2** function is used to enter a PIN2 code.

Requirements
Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class

2.11 DoEnterPUK

The function enters a specified PUK code.

```
bool DoEnterPUK(
  LPCTSTR lpszPUK
);
```

Parameters
lpszPUK
[in] Pointer to a null-terminated string that specifies the PUK code.

Return Values
Nonzero if the **DoEnterPUK** function was successful. Otherwise, a value of **false** is returned, and a specific error code can be retrieved by calling GetLastError.

Remarks
The **DoEnterPUK** function is used to enter a PUK code.

Requirements
Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class

2.12 DoEnterPUK2

The function enters a specified PUK2 code.

```
bool DoEnterPUK2(
  LPCTSTR lpszPUK2
```

```
);
```

Parameters
lpszPUK2
[in] Pointer to a null-terminated string that specifies the PUK2 code.

Return Values
Nonzero if the **DoEnterPIK2** function was successful. Otherwise, a value of **false** is returned, and a specific error code can be retrieved by calling GetLastError.

Remarks
The **DoEnterPUK2** function is used to enter a PUK2 code.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class

2.13 DoEnterServiceCenterAddress

The function enters a specified service center address to modem.

```
bool DoEnterServiceCenterAddress(
  LPCTSTR lpszNumber
);
```

Parameters
lpszNumber
[in] Pointer to a null-terminated string that specifies the service center address.

Return Values
Nonzero if the **DoEnterServiceCenterAddress** function was successful. Otherwise, a value of **false** is returned, and a specific error code can be retrieved by calling GetLastError.

Remarks
The **DoEnterServiceCenterAddress** function is used to enter a service center address.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class

2.14 DoGetSMSIndex

The function retrieves an index which points to the first available stored SMS message.

```
bool DoGetSMSIndex(
  int& nSMSIndex
);
```

Parameters

nSMSIndex

[out] Reference to variable in which function retrieves an index which points to the first available stored SMS message.

Return Values

Nonzero if the **DoGetSMSIndex** function was successful. Otherwise, a value of **false** is returned, and a specific error code can be retrieved by calling GetLastError.

Remarks

The **DoGetSMSIndex** function is used to retrieve an index which points to the first available stored SMS message.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also

CGSMModem class

2.15 DoReadModemAnswer

The function reads an modem answer from the COM port.

```
ModemAnswerEnum DoReadModemAnswer( );
```

Return Values

The following return values are possible for **DoReadModemAnswer** function.

Value	Description
emaCmeError	+CME: ERROR:
emaCmsError	+CMS: ERROR:
emaCmgs	+CMGS:
emaCpinReadyCRLF	+CPIN: READY\r\n
emaCpinSimPinCRLF	+CPIN: SIM PIN\r\n
emaCpinSimPukCRLF	+CPIN: SIM PUK\r\n
emaCpinSimPin2CRLF	+CPIN: SIM PIN2\r\n
emaCpinSimPuk2CRLF	+CPIN: SIM PUK2\r\n
emaCpinPhSimPinCRLF	+CPIN: PH-SIM PIN\r\n
emaCpinPhNetPinCRLF	+CPIN: PH-NET PIN\r\n
emaCpin	+CPIN:
emaCSCATMobileSK	+CSCA: "+421903333000"
emaCSCAOrangeSK	+CSCA: "+421905303303"
emaCSCAO2SK	+CSCA: "+421949909909"
emaCOPSOrangeSK	"23101" MCC/MNC: Orange SK
emaCOPSTMobileSK	"23102" MCC/MNC: T-Mobile SK
emaCOPSO2SK	"23106" MCC/MNC: O2 SK
emaCmgr	+CMGR:
emaCmgl	+CMGL:
emaOKCRLF	OK\r\n
ema0CR	0\r
emaErrorCRLF	ERROR\r\n
emaNoValidAnswerFound	No valid answer found.

Remarks

The **DoReadModemAnswer** function is used to read an modem answer from COM port. The modem answer is returned as the `ModemAnswerEnum` enum. It is availabne in header **ModemAnswer.h**.

Example code

The following code example shows the use of the **DoReadModemAnswer** function.

The function sends the **AT+<CR><LF>** command to the COM port and wait for a modem answer.

```
bool DoAT(void)
{
     // @flow0 | Send the AT command
     CString strATCommand = _T("AT\r");
     if ( DoATCommand(strATCommand) == false )
     {
           return false;
     }

     // @flow0 | Read an answer from modem
     if ( DoReadModemAnswer() == emaOKCRLF )
     {
```

```
            return true;
        }

        return false;
    }
```

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also

CGSMModem class

2.15.1 ModemAnswerEnum

Value	Description
emaCmeError	+CME: ERROR:
emaCmsError	+CMS: ERROR:
emaCmgs	+CMGS:
emaCpinReadyCRLF	+CPIN: READY\r\n
emaCpinSimPinCRLF	+CPIN: SIM PIN\r\n
emaCpinSimPukCRLF	+CPIN: SIM PUK\r\n
emaCpinSimPin2CRLF	+CPIN: SIM PIN2\r\n
emaCpinSimPuk2CRLF	+CPIN: SIM PUK2\r\n
emaCpinPhSimPinCRLF	+CPIN: PH-SIM PIN\r\n
emaCpinPhNetPinCRLF	+CPIN: PH-NET PIN\r\n
emaCpin	+CPIN:
emaCSCATMobileSK	+CSCA: "+421903333000"
emaCSCAOrangeSK	+CSCA: "+421905303303"
emaCSCAO2SK	+CSCA: "+421949909909"
emaCOPSOrangeSK	"23101" MCC/MNC: Orange SK
emaCOPSTMobileSK	"23102" MCC/MNC: T-Mobile SK
emaCOPSO2SK	"23106" MCC/MNC: O2 SK
emaCmgr	+CMGR:
emaCmgl	+CMGL:
emaOKCRLF	OK\r\n
ema0CR	0\r
emaErrorCRLF	ERROR\r\n
emaNoValidAnswerFound	No valid answer found.

2.16 DoReadSMS

The function reads the first available stored SMS message.

```
int DoReadSMS( );
```

Return Values

The following return values are possible for **DoReadSMS** function.

Value	Description
SMS_ERROR	Error occured. If the a value of SMS_ERROR is returned then a specific error code can be retrieved by calling GetLastError.
SMS_RECV_NOSMS	No SMS available for reading.
SMS_RECV_SUBMIT	Ok. A received SMS was read successful.
SMS_RECV_DELIVERY	Ok. A delivery SMS was read successful.

Remarks

The **DoReadSMS** function is used to read stored SMS messages. Each SMS that was read sucessfull is immediately deleted.

Example code

The Sample1 example shows the use of the **DoReadSMS** function.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also

CGSMModem class, _DoReadSMS

2.17 DoSendSMS

The function sends the SMS text to specified phone number.

```
int DoSendSMS(
  LPCTSTR lpszPhoneNumber,
  LPCTSTR lpszSMSText,
  BYTE nNumberOfRetries = 3
);
```

Parameters

lpszPhoneNumber

[in] Pointer to a null-terminated string that specifies the target phone number.

lpszSMSText

[in] Pointer to a null-terminated string that specifies the SMS text to be sent. The maximal SMS text lenght is 160 charactes.

nNumberOfRetries
[in] Specifies the SMS sending retry count.

Return Values
If no error occurs, **DoSendSMS** returns the reference number of sent SMS. Otherwise, a value of SMS_ERROR is returned, and a specific error code can be retrieved by calling GetLastError.

Remarks
The **DoSendSMS** function is used to send the SMS text to specified phone number.

The **SAEAUT SMS Library (MFC) DEMO** version adds to the SMS message prefix www.saeautom.eu/SMSLibrary/ and truncates a new SMS message on valid length (160 characters).

Example code
The Sample1 example shows the use of the **DoSendSMS** function.

Requirements
Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class, _DoSendSMS, Example Code DoSendSMS

2.18 DoSetDefaultConfigurationZ

The function restores the configuration profile.

```
bool DoSetDefaultConfigurationZ( );
```

Return Values
Nonzero if the **DoSetDefaultConfigurationZ** function was successful. Otherwise, a value of **false** is returned, and a specific error code can be retrieved by calling GetLastError.

Remarks
The **DoSetDefaultConfigurationZ** function is used to send a **ATZ+<CR><LF>** command to COM port and wait for a modem answer. The modem answer is evaluated via DoReadModemAnswer function.

Requirements
Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class

2.19 DoSetEcho

The function activates or deactivate the sending ECHO characters in modem answer. The function sends the specific AT command to the COM port and wait for a modem answer.

```
bool DoSetEcho(
  bool bEcho = true
);
```

Parameters

bEcho

[in] Specifies the boolean value to be used. The following values are possible for this member.

Value	Description
true	To activate the ECHO.
false	To deactivate the ECHO.

Return Values

Nonzero if the **DoSetEcho** function was successful. Otherwise, a value of **false** is returned, and a specific error code can be retrieved by calling GetLastError.

Remarks

The **DoSetEcho** function is used to send a **ATE1+<CR><LF>** or **ATE0+<CR><LF>** command to COM port and wait for a modem answer. The modem answer is evaluated via DoReadModemAnswer function.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also

CGSMModem class

2.20 DoSetReportMobileEquipementErrors

The function enables or disables the use of result code.

```
bool DoSetReportMobileEquipmentErrors(
  bool bEnable = true
);
```

Return Values

Nonzero if the **DoSetReportMobileEquipementErrors** function was successful. Otherwise, a value of **false** is returned, and a specific error code can be retrieved by calling GetLastError.

Remarks

The **DoSetReportMobileEquipementErrors** function is used to send a **AT+CMEE=1+<CR><LF>** or **AT+CMEE=0+<CR><LF>** command to COM port and wait for a modem answer. The sending command **+CMEE** disables or enables the use of result code. The modem answer is evaluated via DoReadModemAnswer function.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class

2.21 GetComPortBaudRate

The function retrieves the COM port baudrate parameter.

```
int GetComPortBaudRate( );
```

Return Values
The COM port baudrate parameter if the **GetComPortBaudRate** function was successful. Otherwise, a value of SMS_ERROR is returned.

Remarks
The **GetComPortBaudRate** function is used to retrieves the COM baudrate parameter. For example, when you call **GetComPortName** to a new new instance of CGSMModem class, then CBR_9600 value is returned.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class, SetComPort

2.22 GetComPortDataBits

The function retrieves the COM port databits parameter.

```
int GetComPortDataBits( );
```

Return Values
The COM port databits parameter if the **GetComPortDataBits** function was successful. Otherwise, a value of SMS_ERROR is returned.

Remarks
The **GetComPortDataBits** function is used to retrieves the COM databits parameter.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class, SetComPort

2.23 GetComPortName

The function retrieves the COM port name.

```
CString GetComPortName( );
```

Return Values
The COM port name if the **GetComPortName** function was successful. Otherwise, the zero-lenght string is returned.

Remarks
The **GetComPortName** function is used to retrieve the COM port name. For example, when you call **GetComPortName** to a new new instance of CGSMModem class, then "**COM1**" string value is returned.

Example code
The Sample1 example shows the use of the **GetComPortName** function.

Requirements
Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class, SetComPort

2.24 GetComPortParity

The function retrieves the COM port parity parameter.

```
int GetComPortParity( );
```

Return Values
The COM port parity parameter if the **GetComPortParity** function was successful. Otherwise, a value of SMS_ERROR is returned.

Remarks
The **GetComPortParity** function is used to retrieves the COM parity parameter.

Requirements
Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class, SetComPort

2.25 GetDeliveryStatusText

The function retrieves the SMS delivery status text.

```
static CString GetDeliveryStatusText(
  int nDeliveryStatus
);
```

Parameters

nDeliveryStatus

[in] Specifies the SMS delivery status number code.

Return Values

The SMS delivery status text. The following return values are possible.

Value (Hexadecimal)	Description
0x00	Short message delivered successfully.
0x01	Forwarded, but status unknown.
0x02	Replaced.
0x20	Congestion, still trying.
0x21	Recipient busy, still trying.
0x22	No response recipient, still trying.
0x23	Service rejected, still trying.
0x24	QOS not available, still trying.
0x25	Recipient error, still trying.
0x40	RPC Error.
0x41	Incompatible destination.
0x42	Connection rejected.
0x43	Not obtainable.
0x44	QOS not available.
0x45	No internetworking available.
0x46	Message expired.
0x47	Message deleted by sender.
0x48	Message deleted by SMSC.
0x49	Does not exist.
otherwise	Unknow error.

Remarks

The **GetDeliveryStatusText** function is used to retrieve the SMS delivery status text. The **GetDeliveryStatusText** function relates to the GetRecvSMSDeliveryStatus function.

Example code

The Sample1 example shows the use of the **GetDeliveryStatusText** function.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also

CGSMModem class, GetRecvSMSDeliveryStatus, DoReadSMS, _DoReadSMS, Delivery status codes

2.26 GetErrorMessage

The function retrieves the error message string for an input error code.

```
static DWORD GetErrorMessage(
  DWORD dwErrCode
);
```

Parameters

dwErrCode

[in] Specifies the error code.

Return Values

The return value retrieves the error message string for an input error code. The **GetErrorMessage** function relates to the GetLastError function. The following return values are possible for **GetErrorMessage** function.

General Errors (1-999)

Value	Description
1	COM Port is not available.
2	GSM modem is not available.
3	Writing to COM Port failed.
4	Reading from COM Port failed.
5	GSM modem answer: ERRORCRLF
6	PIN code can't be empty.
7	Enter PUK code.
8	Enter PIN2 code.
9	Enter PUK2 code.
10	Invalid SMS reference number.
11	Invalid SMS index.
12	Invalid SMS length.
13	Timeout occured while reading from COM Port.
14	SMS message is too long.
15	Undefined SMS reference number.
16 - 999 (Not used)	Unknown error.

CME Errors (1000-1999)

Value	Description
1000	Phone failure
1001	No connection to phone
1002	Phone adapter link reserved
1003	Operation not allowed
1004	Operation not supported
1005	PH_SIM PIN required
1006	PH_FSIM PIN required
1007	PH_FSIM PUK required
1010	SIM not inserted
1011	SIM PIN required
1012	SIM PUK required
1013	SIM failure
1014	SIM busy
1015	SIM wrong
1016	Incorrect password
1017	SIM PIN2 required
1018	SIM PUK2 required
1020	Memory full
1021	Invalid index
1022	Not found
1023	Memory failure
1024	Text string too long
1025	Invalid characters in text string
1026	Dial string too long
1027	Invalid characters in dial string
1030	No network service
1031	Network timeout
1032	Network not allowed, emergency calls only
1040	Network personalization PIN required
1041	Network personalization PUK required
1042	Network subset personalization PIN required
1043	Network subset personalization PUK required
1044	Service provider personalization PIN required
1045	Service provider personalization PUK required
1046	Corporate personalization PIN required
1047	Corporate personalization PUK required
1048	PH-SIM PUK required
1100	Unknown error
1103	Illegal MS
1106	Illegal ME

CMS Errors (2000-2999)

Value	Description
2001	Unassigned number
2008	Operator determined barring
2010	Call bared
2021	Short message transfer rejected
2027	Destination out of service
2028	Unindentified subscriber
2029	Facility rejected
2030	Unknown subscriber
2038	Network out of order
2041	Temporary failure
2042	Congestion
2047	Recources unavailable
2050	Requested facility not subscribed
2069	Requested facility not implemented
2081	Invalid short message transfer reference value
2095	Invalid message unspecified
2096	Invalid mandatory information
2097	Message type non existent or not implemented
2098	Message not compatible with short message protocol
2099	Information element non-existent or not implemente
2111	Protocol error, unspecified
2127	Internetworking , unspecified
2128	Telematic internetworking not supported
2129	Short message type 0 not supported
2130	Cannot replace short message
2143	Unspecified TP-PID error
2144	Data code scheme not supported
2145	Message class not supported
2159	Unspecified TP-DCS error
2160	Command cannot be actioned
2161	Command unsupported
2175	Unspecified TP-Command error
2176	TPDU not supported
2192	SC busy
2193	No SC subscription
2194	SC System failure
2195	Invalid SME address
2196	Destination SME barred
2197	SM Rejected-Duplicate SM
2198	TP-VPF not supported

Example code

The Sample1 example shows the use of the **GetErrorMessage** function.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also

CGSMModem class, GetLastError, Error codes

2.27 GetLastError

The function retrieves the error code for the last operation that failed.

```
DWORD GetLastError( );
```

Return Values

The return value indicates the error code for the last operation that failed. The **GetLastError** function relates to the GetErrorMessage function. The following return values are possible for **GetLastError** function.

General Errors (1-999)

Value	Description
1	COM Port is not available.
2	GSM modem is not available.
3	Writing to COM Port failed.
4	Reading from COM Port failed.
5	GSM modem answer: ERRORCRLF
6	PIN code can't be empty.
7	Enter PUK code.
8	Enter PIN2 code.
9	Enter PUK2 code.
10	Invalid SMS reference number.
11	Invalid SMS index.
12	Invalid SMS length.
13	Timeout occured while reading from COM Port.
14	SMS message is too long.
15	Undefined SMS reference number.
16 - 999 (Not used)	Unknown error.

CME Errors (1000-1999)

Value	Description
1000	Phone failure
1001	No connection to phone
1002	Phone adapter link reserved
1003	Operation not allowed
1004	Operation not supported
1005	PH_SIM PIN required
1006	PH_FSIM PIN required
1007	PH_FSIM PUK required
1010	SIM not inserted
1011	SIM PIN required
1012	SIM PUK required
1013	SIM failure
1014	SIM busy
1015	SIM wrong
1016	Incorrect password
1017	SIM PIN2 required
1018	SIM PUK2 required
1020	Memory full
1021	Invalid index
1022	Not found
1023	Memory failure
1024	Text string too long
1025	Invalid characters in text string
1026	Dial string too long
1027	Invalid characters in dial string
1030	No network service
1031	Network timeout
1032	Network not allowed, emergency calls only
1040	Network personalization PIN required
1041	Network personalization PUK required
1042	Network subset personalization PIN required
1043	Network subset personalization PUK required
1044	Service provider personalization PIN required
1045	Service provider personalization PUK required
1046	Corporate personalization PIN required
1047	Corporate personalization PUK required
1048	PH-SIM PUK required
1100	Unknown error
1103	Illegal MS
1106	Illegal ME

CMS Errors (2000-2999)

Value	Description
2001	Unassigned number
2008	Operator determined barring
2010	Call bared
2021	Short message transfer rejected
2027	Destination out of service
2028	Unindentified subscriber
2029	Facility rejected
2030	Unknown subscriber
2038	Network out of order
2041	Temporary failure
2042	Congestion
2047	Recources unavailable
2050	Requested facility not subscribed
2069	Requested facility not implemented
2081	Invalid short message transfer reference value
2095	Invalid message unspecified
2096	Invalid mandatory information
2097	Message type non existent or not implemented
2098	Message not compatible with short message protocol
2099	Information element non-existent or not implemente
2111	Protocol error, unspecified
2127	Internetworking , unspecified
2128	Telematic internetworking not supported
2129	Short message type 0 not supported
2130	Cannot replace short message
2143	Unspecified TP-PID error
2144	Data code scheme not supported
2145	Message class not supported
2159	Unspecified TP-DCS error
2160	Command cannot be actioned
2161	Command unsupported
2175	Unspecified TP-Command error
2176	TPDU not supported
2192	SC busy
2193	No SC subscription
2194	SC System failure
2195	Invalid SME address
2196	Destination SME barred
2197	SM Rejected-Duplicate SM
2198	TP-VPF not supported

Remarks

When a particular member function indicates that an error has occurred, **GetLastError** should be called to retrieve the appropriate error code.

Example code

The Sample1 example shows the use of the **GetLastError** function.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also

CGSMModem class, GetErrorMessage, Error codes

2.27.1 Error codes

General Errors (1-999)

Value	Description
1	COM Port is not available.
2	GSM modem is not available.
3	Writing to COM Port failed.
4	Reading from COM Port failed.
5	GSM modem answer: ERRORCRLF
6	PIN code can't be empty.
7	Enter PUK code.
8	Enter PIN2 code.
9	Enter PUK2 code.
10	Invalid SMS reference number.
11	Invalid SMS index.
12	Invalid SMS length.
13	Timeout occured while reading from COM Port.
14	SMS message is too long.
15	Undefined SMS reference number.
16 - 999 (Not used)	Unknown error.

CME Errors (1000-1999)

Value	Description
1000	Phone failure
1001	No connection to phone
1002	Phone adapter link reserved
1003	Operation not allowed
1004	Operation not supported
1005	PH_SIM PIN required
1006	PH_FSIM PIN required
1007	PH_FSIM PUK required
1010	SIM not inserted
1011	SIM PIN required
1012	SIM PUK required
1013	SIM failure
1014	SIM busy
1015	SIM wrong
1016	Incorrect password
1017	SIM PIN2 required
1018	SIM PUK2 required
1020	Memory full
1021	Invalid index
1022	Not found
1023	Memory failure
1024	Text string too long
1025	Invalid characters in text string
1026	Dial string too long
1027	Invalid characters in dial string
1030	No network service
1031	Network timeout
1032	Network not allowed, emergency calls only
1040	Network personalization PIN required
1041	Network personalization PUK required
1042	Network subset personalization PIN required
1043	Network subset personalization PUK required
1044	Service provider personalization PIN required
1045	Service provider personalization PUK required
1046	Corporate personalization PIN required
1047	Corporate personalization PUK required
1048	PH-SIM PUK required
1100	Unknown error
1103	Illegal MS
1106	Illegal ME

CMS Errors (2000-2999)

Value	Description
2001	Unassigned number
2008	Operator determined barring
2010	Call bared
2021	Short message transfer rejected
2027	Destination out of service
2028	Unindentified subscriber
2029	Facility rejected
2030	Unknown subscriber
2038	Network out of order
2041	Temporary failure
2042	Congestion
2047	Recources unavailable
2050	Requested facility not subscribed
2069	Requested facility not implemented
2081	Invalid short message transfer reference value
2095	Invalid message unspecified
2096	Invalid mandatory information
2097	Message type non existent or not implemented
2098	Message not compatible with short message protocol
2099	Information element non-existent or not implemente
2111	Protocol error, unspecified
2127	Internetworking , unspecified
2128	Telematic internetworking not supported
2129	Short message type 0 not supported
2130	Cannot replace short message
2143	Unspecified TP-PID error
2144	Data code scheme not supported
2145	Message class not supported
2159	Unspecified TP-DCS error
2160	Command cannot be actioned
2161	Command unsupported
2175	Unspecified TP-Command error
2176	TPDU not supported
2192	SC busy
2193	No SC subscription
2194	SC System failure
2195	Invalid SME address
2196	Destination SME barred
2197	SM Rejected-Duplicate SM
2198	TP-VPF not supported

2.28 GetModemManufacturer

The function retrieves the modem manufacturer identification.

```
CString GetModemManufacturer( );
```

Return Values
The modem manufacturer identification.

Remarks
The **GetModemManufacturer** function is used to send a **AT+CGMI+<CR><LF>** command to COM port and wait for a modem answer. The modem answer is evaluated via DoReadModemAnswer function.

Example code
The Sample1 example shows the use of the **GetModemManufacturer** function.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class, GetModemModel

2.29 GetModemModel

The function retrieves the modem model identification.

```
CString GetModemModel( );
```

Return Values
The modem model identification.

Remarks
The **GetModemModel** function is used to send a **AT+CGMM+<CR><LF>** command to COM port and wait for a modem answer. The modem answer is evaluated via DoReadModemAnswer function.

Example code
The Sample1 example shows the use of the **GetModemModel** function.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class, GetModemManufacturer

2.30 GetModemReadBuffer

The function retrieves the modem read buffer data in string format.

```
CString GetModemReadBuffer( );
```

Return Values

The modem read buffer data in string format if the **GetModemReadBuffer** function was successful. Otherwise, the zero-lenght string is returned, and a specific error code can be retrieved by calling GetLastError.

Remarks

The **GetModemReadBuffer** function is used to retrieve the modem read buffer data in string format. The **GetModemReadBuffer** function relates to the DoReadModemAnswer function.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also

CGSMModem class, DoReadModemAnswer, DoReadSMS, _DoReadSMS

2.31 GetRecvSMSDeliveryRefNumber

The function retrieves the SMS reference number for the last received SMS message.

```
int GetRecvSMSDeliveryRefNumber( );
```

Return Values

The SMS reference number.

Remarks

The **GetRecvSMSDeliveryRefNumber** function is used to retrieve the SMS reference number for the last received SMS message. The reference number makes a relation with to reference number returned from DoReadSMS function or _DoSendSMS function. The **GetRecvSMSDeliveryRefNumber** function relates to the DoReadSMS function and the _DoReadSMS function. The **GetRecvSMSDeliveryRefNumber** function is possible use only for SMS_RECV_DELIVERY message type.

Example code

The Sample1 example shows the use of the **GetRecvSMSDeliveryRefNumber** function.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also

CGSMModem class, DoReadSMS, _DoReadSMS

2.32 GetRecvSMSDeliveryStatus

The function retrieves the delivery status code for the last received SMS message.

```
int GetRecvSMSDeliveryStatus( );
```

Return Values

The delivery status code. The following return values are possible.

Value (Hexadecimal)	Description
0x00	Short message delivered successfully.
0x01	Forwarded, but status unknown.
0x02	Replaced.
0x20	Congestion, still trying.
0x21	Recipient busy, still trying.
0x22	No response recipient, still trying.
0x23	Service rejected, still trying.
0x24	QOS not available, still trying.
0x25	Recipient error, still trying.
0x40	RPC Error.
0x41	Incompatible destination.
0x42	Connection rejected.
0x43	Not obtainable.
0x44	QOS not available.
0x45	No internetworking available.
0x46	Message expired.
0x47	Message deleted by sender.
0x48	Message deleted by SMSC.
0x49	Does not exist.
otherwise	Unknow error.

Remarks

The **GetRecvSMSDeliveryStatus** function is used to retrieve the delivery status code for the last received SMS message. The **GetRecvSMSDeliveryStatus** function relates to the DoReadSMS function and the _DoReadSMS function. The **GetRecvSMSDeliveryStatus** function is possible use only for SMS_RECV_DELIVERY message type.

Example code

The Sample1 example shows the use of the **GetRecvSMSDeliveryStatus** function.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also

CGSMModem class, DoReadSMS, _DoReadSMS, Delivery status codes

2.32.1 Delivery status codes

Value (Hexadecimal)	Description
0x00	Short message delivered successfully.
0x01	Forwarded, but status unknown.
0x02	Replaced.
0x20	Congestion, still trying.
0x21	Recipient busy, still trying.
0x22	No response recipient, still trying.
0x23	Service rejected, still trying.
0x24	QOS not available, still trying.
0x25	Recipient error, still trying.
0x40	RPC Error.
0x41	Incompatible destination.
0x42	Connection rejected.
0x43	Not obtainable.
0x44	QOS not available.
0x45	No internetworking available.
0x46	Message expired.
0x47	Message deleted by sender.
0x48	Message deleted by SMSC.
0x49	Does not exist.
otherwise	Unknow error.

2.33 GetRecvSMSSenderPhoneNumber

The function retrieves the sender phone number for the last received SMS message.

```
CString GetRecvSMSSenderPhoneNumber( );
```

Return Values

The sender phone number if the **GetRecvSMSSenderPhoneNumber** function was successful. Otherwise, the zero-lenght string is returned.

Remarks

The **GetRecvSMSSenderPhoneNumber** function is used to retrieve the sender phone number for the last received SMS message. The **GetRecvSMSSenderPhoneNumber** function relates to the DoReadSMS function and the _DoReadSMS function. The **GetRecvSMSSenderPhoneNumber** function is possible use for both SMS_RECV_SUBMIT and SMS_RECV_DELIVERY message types.

Example code

The Sample1 example shows the use of the **GetRecvSMSSenderPhoneNumber** function.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also

CGSMModem class, DoReadSMS, _DoReadSMS

2.34 GetRecvSMSServiceCentreTimestamp

The function retrieves the service centre timestamp for the last received SMS message.

```
CString GetRecvSMSServiceCentreTimestamp( );
```

Return Values

The SMS message service center timestamp if the **GetRecvSMSTimestamp** function was successful. Otherwise, the zero-lenght string is returned.

Remarks

The **GetRecvSMSServiceCentreTimestamp** function is used to retrieve the service centre timestamp for the last received SMS message. The **GetRecvSMSServiceCentreTimestamp** function relates to the DoReadSMS function and the _DoReadSMS function. The **GetRecvSMSServiceCentreTimestamp** function is possible use only for SMS_RECV_DELIVERY message type.

Example code

The Sample1 example shows the use of the **GetRecvSMSServiceCentreTimestamp** function.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also

CGSMModem class, DoReadSMS, _DoReadSMS

2.35 GetRecvSMSText

The function retrieves the SMS text for the last received SMS message.

```
CString GetRecvSMSText( );
```

Return Values

The SMS message text.

Remarks

The **GetRecvSMSText** function is used to retrieve the SMS text for the last received SMS message. The **GetRecvSMSText** function relates to the DoReadSMS function and the _DoReadSMS function. The **GetRecvSMSText** function is possible use only for SMS_RECV_SUBMIT message type.

Example code

The Sample1 example shows the use of the **GetRecvSMSText** function.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also

CGSMModem class, DoReadSMS, _DoReadSMS

2.36 GetRecvSMSTimestamp

The function retrieves the timestamp for the last received SMS message.

```
CString GetRecvSMSTimestamp( );
```

Return Values

The SMS message timestamp if the **GetRecvSMSTimestamp** function was successful. Otherwise, the zero-lenght string is returned.

Remarks

The **GetRecvSMSTimestamp** function is used to retrieve the timestamp for the last received SMS message. The **GetRecvSMSTimestamp** function relates to the DoReadSMS function and the _DoReadSMS function. The **GetRecvSMSTimestamp** function is possible use for both SMS_RECV_SUBMIT and SMS_RECV_DELIVERY message types.

Example code

The Sample1 example shows the use of the **GetRecvSMSTimestamp** function.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also

CGSMModem class, DoReadSMS, _DoReadSMS

2.37 SetComPort

The function only initializes the COM port communication parameters in CGSMModem class object instance.

```
void SetComPort(
  LPCTSTR lpszPortName,
  int nBaudRate = CBR_9600,
  int nDataBits = 8,
  int nStopBits = ONESTOPBIT,
  int nParity = NOPARITY
);
```

Parameters

lpszPortName
[in] Pointer to a null-terminated string that specifies the name of COM port.

nBaudRate
[in] Specifies the baud rate at which the communication device operates. It is an actual baud rate value, or one of the following baud rate indexes:

- CBR_110
- CBR_300
- CBR_600
- CBR_1200
- CBR_2400
- CBR_4800
- CBR_9600
- CBR_14400
- CBR_19200
- CBR_38400
- CBR_56000
- CBR_57600
- CBR_115200
- CBR_128000
- CBR_256000

nDataBits
[in] Specifies the number of bits in the bytes transmitted and received.

nStopBits
[in] Specifies the number of stop bits to be used. The following values are possible for this member.

Value	Description
ONESTOPBIT	1 stop bit
ONE5STOPBITS	1.5 stop bits
TWOSTOPBITS	2 stop bits

nParity
[in] Specifies the parity scheme to be used. The following values are possible for this member.

Value	Description
EVENPARITY	Even
MARKPARITY	Mark
NOPARITY	No parity
ODDPARITY	Odd
SPACEPARITY	Space

Remarks

The **SetComPort** function is used to initialize the COM port communication parameters in CGSMModem class object instance. The **SetComPort** function relates to the Connect function.

Example code

The Sample1 example shows the use of the **SetComPort** function.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class, Connect

2.38 SetPIN

The function only initializes the PIN code parameter in CGSMModem class object instance.

```
void SetPIN(
  LPCTSTR lpszPIN
);
```

Parameters
lpszPIN
[in] Pointer to a null-terminated string that specifies the PIN code.

Remarks
The **SetPIN** function is used to initialize the PIN code parameter in CGSMModem class object instance. The **SetPIN** function relates to the DoEnterPIN function.

Example code
The Sample1 example shows the use of the **SetPIN** function.

Requirements
Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class, DoEnterPIN

2.39 SetReadTimeout

The function only initializes the **ReadTimeout** communication parameter in CGSMModem class object instance. It specifies the maximal number of mili-seconds to wait for modem answer.

```
void SetReadTimeout(
  DWORD dwTimeout = 10000
);
```

Parameters
dwTimeout
[in] Specifies the maximal number of mili-seconds to wait for modem answer.

Remarks
The **SetReadTimeout** function is used to initialize the **ReadTimeout** communication parameter in CGSMModem class object instance. It specifies the maximal number of mili-seconds to wait for modem answer. The default value is 10 seconds.

Example code
The Sample1 example shows the use of the **SetReadTimeout** function.

Requirements
Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class, SetComPort

2.40 SetServiceCenterAddress

The function only initializes the service center address parameter in CGSMModem class object instance.

```
void SetServiceCenterAddress(
  LPCTSTR lpszNumber
);
```

Parameters
lpszNumber
[in] Pointer to a null-terminated string that specifies the service center address.

Remarks
The **SetServiceCenterAddress** function is used to initialize the service center address parameter in CGSMModem class object instance. The **SetServiceCenterAddress** function relates to the DoEnterServiceCenterAddress function.

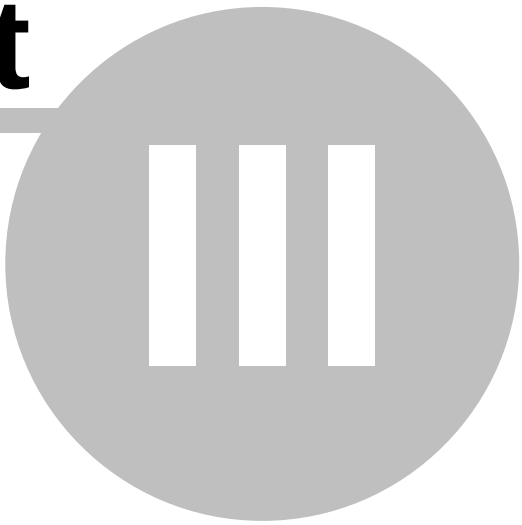
Example code
The Sample1 example shows the use of the **SetServiceCenterAddress** function.

Requirements
Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also
CGSMModem class, DoEnterServiceCenterAddress

SAEAUT SNMP OPC Server

Part



3 Sample Applications

The sample code supplied with this SAEAUT SMS Library (MFC) is in the form of projects for **Microsoft Visual Studio.NET 2005**. The samples are in C++.

The following samples are installed by default in *C:\Program Files\SAE - Automation, s.r.o\SMS Library\MFC\samples*.

Usage information for each sample is contained in a readme.txt file in each sample directory.

Samples

Name	Description
Sample1	The purpose of this sample is to show our customers how to easy use SAEAUT SMS Library (MFC) for sending and receiving of SMS messages from their applications (source codes). The sample is C++ application that shows how to send and read SMS messages via associated GSM modem using the SAEAUT SMS Library (MFC). It demonstrates the use of the CGSMModem class methods that can be used from customer application code. The sample requires the library **SMSLib.lib** to build.

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also

SAEAUT SMS Library (MFC), CGSMModem class

3.1 Sample1

The purpose of this sample is to show our customers how to easy use SAEAUT SMS Library (MFC) for sending and receiving of SMS messages from their applications (source codes).

Remarks

The **Sample1** sample is C++ application that shows how to send and read SMS messages via associated GSM modem using the SAEAUT SMS Library (MFC). It demonstrates the use of the CGSMModem class methods that can be used from customer application code.

Source code

```
int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
{
     int nRetCode = 0;

     // initialize MFC and print and error on failure
     if (!AfxWinInit(::GetModuleHandle(NULL), NULL, ::GetCommandLine(),
0))
     {
```

```
            _tprintf(_T("Fatal Error: MFC initialization failed\n"));
            nRetCode = 1;
      }
      else
      {
            _tprintf( _T("*************************************\n") );
            _tprintf( _T("* SAEAUT SMS Library (MFC) 1.0.0      *\n") );
            _tprintf( _T("*  sample of using                    *\n") );
            _tprintf( _T("*************************************\n\n") );

            CGSMModem GSMModem;                              // GSM modem
instance

            // TODO: change all folowing parameters to suit your needs
            CString strComPort  = _T("COM17");               // Com Port
            DWORD dwReadTimeout = 15000;                     // 15 seconds

            GSMModem.SetComPort(strComPort, 9600, 8, 0, 0);
            GSMModem.SetReadTimeout(dwReadTimeout);

            _tprintf( _T("COM Port: %s\n"), GSMModem.GetComPortName() );

            _tprintf( _T("\n\n* Connection to modem in progress, please
wait ...") );

            // make a modem connection
            if (GSMModem.Connect())
            {
                  _tprintf( _T("OK\n\n") );
                  _tprintf( _T("Modem: %s %s\n\n"),
GSMModem.GetModemModel(), GSMModem.GetModemManufacturer() );

                  // TODO: change the PIN code to suit your needs
                  CString strPIN = _T("0000");               // PIN code
e.g. 0000
                  GSMModem.SetPIN(strPIN);
                  _tprintf( _T("PIN: %s\n"), strPIN );


                  // TODO: change the service center address to suit your
needs
                  CString strSCA = _T("+421905303303");      // Service
Center Address e.g. ORANGE SVK (+421905303303)
                  GSMModem.SetServiceCenterAddress(strSCA);
                  _tprintf( _T("SCA: %s\n"), strSCA );


                  // TODO: change the phone number to suit your needs
                  CString strPhoneNumber = _T("XXXX");       // Phone number
e.g 0123456789
                  _tprintf( _T("Phone number: %s\n"), strPhoneNumber );


                  // TODO: change the SMS text to suit your needs
                  CString strSMSText = _T("Hello world!");  // SMT text
                  _tprintf( _T("SMS text: %s\n"), strSMSText );

                  _tprintf( _T("\n\n* Sending SMS in progress, please wait
...\n") );
```

```
                // send the SMS
                int nRefNumber = GSMModem.DoSendSMS( strPhoneNumber,
strSMSText,
3);

                if ( nRefNumber != SMS_ERROR )
                {
                      _tprintf( _T("SMS was sent sucessfully (Ref Nr.
%d).\n\n"), nRefNumber);
                }
                else
                {
                      int nLastError =  GSMModem.GetLastError();

                      _tprintf( _T("Error(%d): %s\n\n"), nLastError,
CGSMModem::GetErrorMessage(nLastError) );
                }


                _tprintf( _T("* Reading SMS in progress, please wait
...\n") );
                Sleep(1000);

                // read SMS
                switch (GSMModem.DoReadSMS())
                {
                case SMS_RECV_NOSMS:
                      {
                            _tprintf( _T("No SMS available for
reading.\n") );
                      }
                      break;

                case SMS_RECV_SUBMIT:
                      {
                            _tprintf( _T("Received SMS: %s %s %s\n"),
                                                                  GSMModem.
GetRecvSMSSenderPhoneNumber(),
                                                                  GSMModem.
GetRecvSMSTimestamp(),
                                                                  GSMModem.
GetRecvSMSText());
                      }
                      break;

                case SMS_RECV_DELIVERY:
                      {
                            int nDeliveryStatus =
GSMModem.GetRecvSMSDeliveryStatus();

                            _tprintf( _T("Delivered SMS (Ref Nr. %d): %s
%s %s (%d) %s\n"),
                                                                  GSMModem.
GetRecvSMSDeliveryRefNumber(),
                                                                  GSMModem.
GetRecvSMSSenderPhoneNumber(),
                                                                  GSMModem.
GetRecvSMSTimestamp(),
```

```
                                                                  GSMModem.
GetRecvSMSServiceCentreTimestamp(),
                                                                  nDelivery
Status,
                                                                  CGSMModem
::GetDeliveryStatusText( nDeliveryStatus ) );
                        }
                        break;

                case SMS_ERROR:
                        {
                                int nLastError =  GSMModem.GetLastError();

                                _tprintf( _T("Error(%d): %s\n"), nLastError,
CGSMModem::GetErrorMessage(nLastError) );
                        }
                }
        }
        else
        {
                _tprintf( _T(" ERROR\n") );

                _tprintf( _T("Error(%d): %s\n"), GSMModem.GetLastError(),
CGSMModem::GetErrorMessage(GSMModem.GetLastError()) );
        }

        // close a modem connection
        GSMModem.Disconnect();
    }

    _tprintf( _T("\n\n* Please wait (15 s) for application terminating
...\n") );

    ::Sleep(15000);

    return nRetCode;
}
```

Requirements

Header	Declared in GSMModem.h.
Library	Link to SMSLib.lib.
DLL	Requires SMSLib.dll

See Also

SAEAUT SMS Library (MFC), CGSMModem class

Part IV

4 Known issues

While working with SAEAUT SMS Library (MFC) you may encounter some problems. Some general comments and solutions are provided below.

1. Incorrect functionality while using USB - RS232 converter

Nowadays, more and more devices with in-built GSM modem provide only USB interface for communication with other device. Depending on the modem, it is required to install particular drivers. After installation of such device a new serial port appears in the device list that is just a virtual port.

After installation of such device it is recommended to **reboot the computer** because of driver initialization.

2. Incorrect functionality while using mobile phone data cable

A mobile phone is a device with in-built GSM modem. If the user wants to use a mobile phone, it may happen that the application will not work correctly. The reason may be the data cable that is designed for data transfer just between the phone and a personal computer.

If such situation occurs, try to change the data cable or try to use other communication interface (e.g. IrDA, Bluetooth, etc.)

3. The application was sending SMS, but now it doesn't work

This problem may occur when a prepaid SIM card is used and the user ran out of limit.

In this case check your SIM card credit balance by inserting the SIM card into a mobile phone and then evaluating the remaining balance by mobile phone's services. If the credit balance is sufficient, try to send a test SMS directly from your mobile phone. If the sending of SMS fails, please contact your mobile network operator.

See Also

CGSMModem class, SAEAUT SMS Library (MFC)

Part V

5 Support & Contact

Support

tel.: +421 (0) 42 44 507 01 (Monday - Friday : 8:00 AM - 17:00 PM)
fax: +421 (0) 42 44 507 02
e-mail: sae-automation@saeautom.sk

Contact

SAE - Automation, s.r.o.
Sady Cyrila a Metoda 21/18
018 51 Nova Dubnica, Slovak republic

http://www.saeautom.eu, sae-automation@saeautom.sk

See Also

SAEAUT SMS Library (MFC)

Index

- C -

CGSMModem class 8, 11, 13, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27, 28, 30, 34, 38, 42, 43, 44, 45, 46, 47, 49, 50, 52
- Connect 13
- constructor 8, 13
- Delivery status codes 28, 44, 45
- destructor 8
- Disconnect 15
- DoATCommand 16
- DoDeleteIndex 17
- DoEnterPIN2 18
- DoEnterPUK 19
- DoEnterPUK2 19
- DoEnterServiceCenterAddress 20
- DoGetSMSIndex 20
- DoReadModemAnswer 21, 23
- DoReadSMS 11, 24
- DoSendSMS 11, 24
- DoSetDefaultConfigurationZ 25
- DoSetEcho 25
- DoSetReportMobileEquipementErrors 26
- Error Codes 30, 34, 38
- GetComPortBaudRate 27
- GetComPortDataBits 27
- GetComPortName 27
- GetComPortParity 28
- GetErrorMessage 30
- GetLastError 34
- GetModemManufacturer 42
- GetModemModel 42
- GetModemReadBuffer 43
- GetRecvSMSDeliveryRefNumber 43
- GetRecvSMSDeliveryStatus 28, 44
- GetRecvSMSSenderPhoneNumber 45
- GetRecvSMSServiceCentreTimestamp 46
- GetRecvSMSText 46
- GetRecvSMSTimestamp 47
- SetComPort 47
- SetPIN 49
- SetReadTimeout 49
- SetServiceCenterAddress 50, 52

Connect 13
constructor 8, 13
contact 59

- D -

Delivery status codes 28, 44, 45
destructor 8
Disconnect 15
DoATCommand 16
DoDeleteIndex 17
DoEnterPIN2 18
DoEnterPUK 19
DoEnterPUK2 19
DoEnterServiceCenterAddress 20
DoGetSMSIndex 20
DoReadModemAnswer 21, 23
DoReadSMS 11, 24
DoSendSMS 11, 24
DoSetDefaultConfigurationZ 25
DoSetEcho 25
DoSetReportMobileEquipementErrors 26

- E -

Error codes 30, 34, 38

- G -

GetComPortBaudRate 27
GetComPortDataBits 27
GetComPortName 27
GetComPortParity 28
GetErrorMessage 30
GetLastError 34
GetModemManufacturer 42
GetModemModel 42
GetModemReadBuffer 43
GetRecvSMSDeliveryRefNumber 43
GetRecvSMSDeliveryStatus 28, 44
GetRecvSMSSenderPhoneNumber 45
GetRecvSMSText 46
GetRecvSMSTimestamp 46, 47

- K -

known issues 57

- M -

main features 4, 5
ModemAnswerEnum 23

- R -

read 11, 21, 23, 24
read SMS 11, 21, 23, 24

- S -

sample applications 52
samples 52
send 11, 24
send SMS 11, 24
SetComPort 47
SetPIN 49
SetReadTimeout 49
SetServiceCenterAddress 50, 52
support 59

Endnotes 2... (after index)