



## HsSmsDII Library User Manual (v1.1)

HsSmsDII Library User Manual (v1.1)	1
1 Introduction	2
1.1 About HsSmsDII	2
1.2 HsSmsDII Architecture	2
2 API specification	3
2.1 C# .NET Interface	3
2.1.1 HsSmsDIIOpenCs	3
2.1.2 HsSmsDIICloseCs	4
2.1.3 HsSmsDIISendCs	5
2.1.4 HsSmsDIIReadCs	7
2.1.5 HsSmsDIIDeleteCs	8
2.1.6 HsSmsDIIReadEventsCs	9
2.1.7 HsSmsDIIGetErrCs	16
2.1.8 HsSmsDIIGetSmscCS	18
2.1.9 HsSmsDIIReadSmsc	18
2.2 Visual C / C++ (v6 and 2003) Interface	19
2.2.1 HsSmsDIIOpen	19
2.2.2 HsSmsDIIClose	20
2.2.3 HsSmsDIISend	21
2.2.4 HsSmsDIIRead	24
2.2.5 HsSmsDIIDelete	26
2.2.6 HsSmsDIIGetSmsc	27
2.2.7 HsSmsDIIExErrorStr	27
2.2.8 Callback Function	28
2.3 Visual Basic (v6 and 2003) Interface	33
2.3.1 HsSmsDIIOpenVB	33
2.3.2 HsSmsDIICloseVB	34
2.3.3 HsSmsDIISendVB	35
2.3.4 HsSmsDIIReadVB	37
2.3.5 HsSmsDIIDelete	40
2.3.6 HsSmsDIIReadEventsVB	41
2.3.7 HsSmsDIIGetErrVB	48
2.4 ActiveX COM Object Interface	49
2.4.1 SmsOpen	49
2.4.2 SmsSend	50
2.4.3 SmsReadEvents	52
2.4.4 SmsRead	57
2.4.5 SmsDelete	58
2.4.6 SmsClose	59
2.4.7 SmsGetRcString	60
2.5 Error Codes and Other Constants	60
2.5.1 HsSmsDII return codes	60
2.5.2 HsSmsDII event notifications	60
2.5.3 HsSmsDII Extra Error Codes	61
2.5.4 Other Constants	61

# 1 Introduction

## 1.1 About HsSmsDll

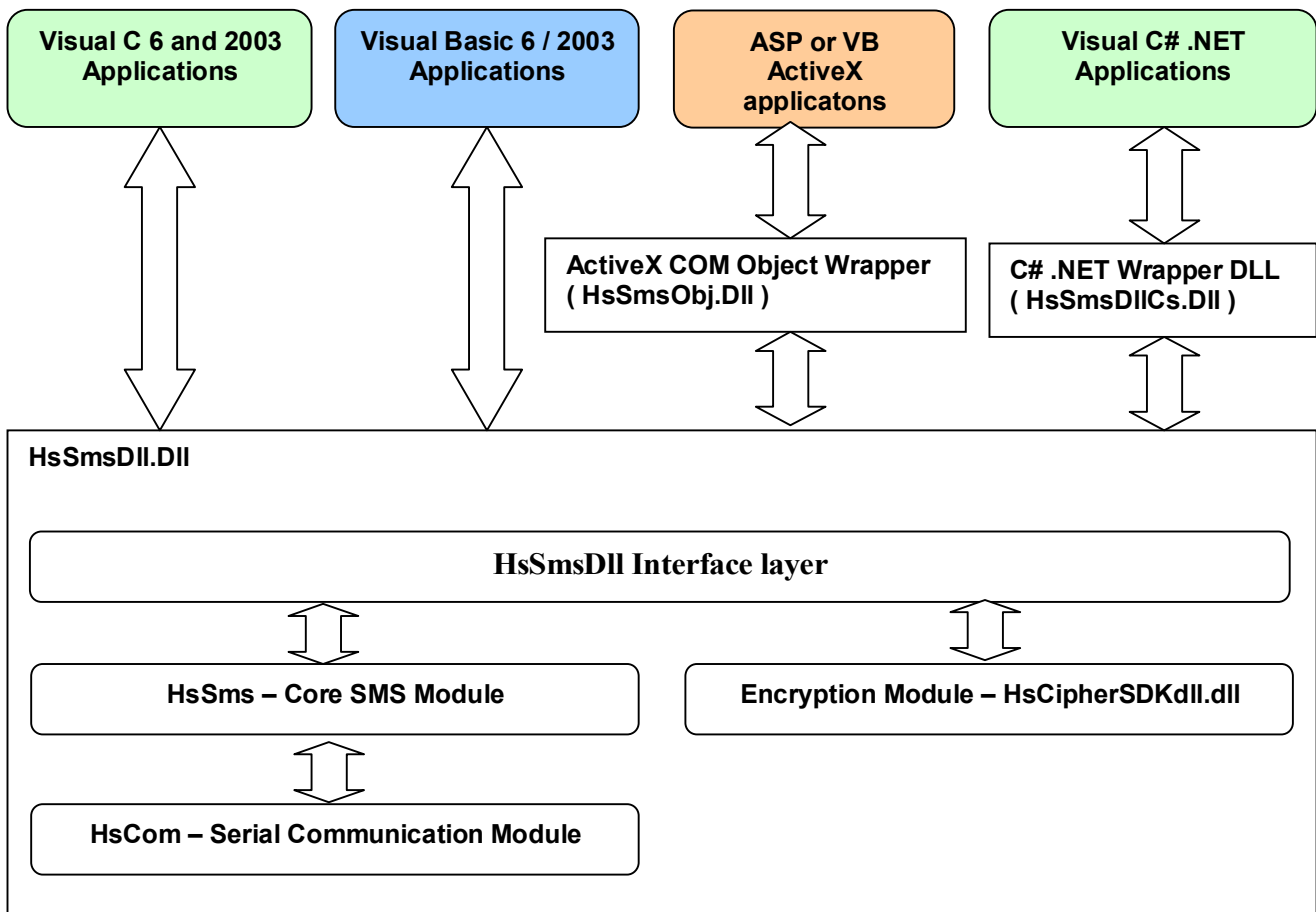
HS SMS DLL is a Windows Dynamic Link Library enabling applications to send, receive, read and delete SMS messages via a GSM modem attached to PC COM port, supporting selected features of standards GSM 07.05 (ETS 300 585) and GSM 03.40.

HS SMS DLL includes integrated encryption which can be optionally used to encrypt and decrypt SMS text using AES, DES, 3DES, ARC4, CAST128, Blowfish or Twofish algorithms.

HS SMS DLL is designed for use from Visual C / C++, VB (v6 and 2003), C# .NET and other programming languages, capable of calling DLL functions

## 1.2 HsSmsDll Architecture

HsSmsDll library internally consists of upper interface layer, underlying core SMS module (HsSms) and encryption module (HsCipherSdkdll.dll). HsSms at the bottom layer interfaces to serial communications module HsCom. HsSmsDll library may be used from both non .NET and .NET applications. If used from C# .NET application, the interface is implemented via an additional C# wrapper DLL HsSmsDllCs.dll. If used from ActiveX applications (ASP or VB), the interface is implemented via an additional COM object wrapper HsSmsObj.dll



## 2 API specification

### 2.1 C# .NET Interface

#### 2.1.1 HsSmsDIIOpenCs

##### Declaration:

```
[DllImport("HsSmsDII.dll")]
public static extern int HsSmsDIIOpenCs(
    int    portno,        // COM port number
    int    baudrate,     // baud rate
    int    databits,     // databits constant
    int    parity,       // parity constant
    int    stopbits,     // stopbits constant
    int    use_fc,       // use hardware flow control
    String pin,          // SIM PIN, may be 0 if not used
    int    init_wait_secs, // hardware initialization delay in seconds
    int    enable_log    // 1=log events to file HsSms.log. 0=no logging
);
```

##### Summary:

Use this method to open and configure a GSM device attached to specified serial port. This function must be called prior to calling any other library functions.

##### Parameters:

###### *portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

###### *baudrate*

– Serial port rate in bauds, one of the following: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200, 128000, 256000

###### *databits*

– Number of data bits, one of the following: 5, 6, 7, 8

###### *parity*

– Parity code as follows: 2 - EVEN, 3 - MARK, 0 - NONE, 1 - ODD, 4 - SPACE

###### *stopbits*

– Stop bits code as follows: 0 – 1 Stop bit, 1 – 1.5 Stop bits, 2 – 2 Stop bits

###### *use\_fc*

– Set to 1 to use hardware RTS/CTS flow control

###### *pin*

– 4 digit SIM PIN in zero terminated ASCII string format. Set to NULL if your SIM does not require PIN.

###### *init\_wait\_secs*

– GSM hardware initialisation delay in seconds. This is the delay between asserting DTR on the serial port being opened and sending first AT configuration command to the GSM device. This delay may be necessary to allow a GSM module to initialise, before it accepts AT commands. Delay value depends on GSM module vendor. If delay is not required, set to 0.

*enable\_log*

- 1=log events to file HsSms.log. 0=no logging

**Return values:**

HS_SMS_RC_OK	Success, GSM device configuration is in progress, the actual result of operation is determined by polling with method <b>HsSmsDllReadEventsCs</b>
HS_SMS_RC_BAD_PORT	Invalid port number
HS_SMS_RC_INV_PAR	Invalid parameter
HS_SMS_RC_PORT_ALR_OPEN	Port is already open
HS_SMS_RC_NO_MEM	No free memory
HS_SMS_RC_COM_OPEN_FAIL	Failed to open COM port

**Sample usage:**

```
using HsSmsDllCs;
```

```
rc = HsSms.HsSmsDllOpenCs(0, 115200, 8, 0, 0, 1, "", 2);
```

## 2.1.2 HsSmsDllCloseCs

**Declaration:**

```
[DllImport("HsSmsDll.dll")]  
public static extern long HsSmsDllCloseCs(  
    int portno // COM port number, zero based  
);
```

**Summary:**

Use this function to close serial port connected to a GSM device and release communications session and resources associated with it.

**Parameters:**

*portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

**Return values:**

HS_SMS_RC_OK	Success, port closed
HS_SMS_RC_BAD_PORT	Invalid port number
HS_SMS_RC_PORT_NOT_OPEN	Specified port is not open

**Sample usage:**

```
using HsSmsDllCs;
```

```
HsSms.HsSmsDllCloseCs(cur_port);
```

### 2.1.3 HsSmsDII SendCs

**Declaration:**

```
[DllImport("HsSmsDII.dll")]
public static extern int HsSmsDII SendCs(
    int portno, // COM port number, zero based
    String smsc_number, // SMS center number
    String dest_number, // destination number (null terminated string)
    StringBuilder sms_text, // message text
    int sms_text_len, // message text length
    int text_encoding, // message encoding (plain ascii or binary as Ascii Hex)
    int is_encryption_on, // 1= encryption enabled, 0=encryption disabled
    int encryption_alg, // encryption algorithm
    StringBuilder encryption_key, // encryption key
    int encrypt_key_bits // encryption key size in bits
);
```

**Summary:**

Use this function to send SMS message. The port must be previously open with function HsSmsDII OpenCs.

**Parameters:**

*portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

*smsc\_number*

– SMS Service Centre number via which the SMS message is to be sent, this number is different for each GSM network operator. The number is of String type

*dest\_number*

– Destination GSM number where the SMS message is to be sent. The number is of String type

*sms\_text*

- SMS message buffer to be sent. The buffer is passed to C# wrapper DLL as StringBuilder type. The HsSmsDII.dll receives the buffer as ANSI byte array. Depending on selected encoding (see parameter text\_encoding) the content of the buffer may be ASCII printable characters or non ASCII binary content.

*sms\_text\_len*

- length of SMS message buffer in bytes.

*text\_encoding*

- This parameter determines how the content of sms\_text buffer is encoded before it is sent to GSM module. This parameter can take of the following values:

SMSENC_ASCII	(0x0) The buffer sms_text is sent “as is” unmodified.
SMSENC_BINHEX	(0x1) Each byte from sms_text buffer is sent as 2 ASCII hex characters. For example the sms_text buffer contains the following binary data {0x01,0xFF,0xEB}. This buffer shall be encoded and sent as follows: “01FFEB” (string) or in hex representation 0x30,0x31,0x46,0x46,0x45,0x42  Using this method the maximum length of sms_text buffer must not exceed 80 bytes

#### *is\_encryption\_on*

- Set to 1 to encrypt the content of sms\_text buffer before sending. The text encoding shall be automatically forced to SMSENC\_BINHEX, regardless of the value of the *text\_encoding* parameter. The maximum length of source sms\_text buffer must not exceed 80 bytes. Set to 0 to send SMS text without encryption.

#### *encryption\_alg*

- If is\_encryption\_on is set to 1, this parameter specifies the encryption algorithm to be used, one of the following:

- 0 – AES
- 1 – DES
- 2 – 3DES
- 3 – ARC4,
- 4 – CAST128
- 5 – BLOWFISH
- 6 – TWOFISH

#### *encryption\_key*

- If is\_encryption\_on is set to 1, this parameter is the buffer of type StringBuilder containing encryption key.

#### *encrypt\_key\_bits*

- If is\_encryption\_on is set to 1, this parameter specifies the length of encryption key in number of bits

#### **Return values:**

HS_SMS_RC_OK	Success, SMS message is submitted for sending. The actual send result is determined by polling with method <b>HsSmsDllReadEventsCs</b>
HS_SMS_RC_NOT_INIT	HsSmsDll library not initialised
HS_SMS_RC_BAD_PORT	Invalid port number
HS_SMS_RC_PORT_NOT_OPEN	Specified port is not open
HS_SMS_RC_INV_STATE	Invalid internal state for sending SMS
HS_SMS_RC_INV_PAR	Invalid parameters

#### **Sample usage:**

```
using HsSmsDllCs;
```

```
StringBuilder cb = new StringBuilder(textBoxSms.Text);  
StringBuilder key = new StringBuilder(textBoxKey.Text);
```

```
rc = HsSms.HsSmsDllSendCs(  
    cur_port, // port  
    textBoxSmsc.Text, // SMS service centre number  
    textBoxNumber.Text, // destination number  
    cb, cb.Length, // SMS text and SMS text length  
    comboBoxEnc.SelectedIndex, // encoding  
    (checkBoxCrypt.Checked == true) ? 1 : 0, // encryption  
    comboBoxAlg.SelectedIndex, // encryption algorithm  
    key, // encryption key  
    key_bitsize // key size in bits  
);
```

## 2.1.4 HsSmsDIIReadCs

### Declaration:

```
[DllImport("HsSmsDII.dll")]
public static extern int HsSmsDIIReadCs(
    int portno,                // COM port number, zero based
    int msg_index,            // index of message to read
    int text_encoding,        // message encoding (plain ascii or binary as Ascii Hex)
    int is_encryption_on,    // 1= decryption enabled, 0=decryption disabled
    int decryption_alg,      // encryption algorithm
    StringBuilder decryption_key, // decryption key
    int decrypt_key_bits     // decryption key size in bits
);
```

### Summary:

Use this function to read an already received SMS message from GSM device, given the message index of the message in mobile device storage.

### Parameters:

#### *portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

#### *msg\_index*

– Message index of SMS message to read in the mobile device storage.

#### *text\_encoding*

- This parameter determines how the content of sms buffer is decoded after it read from mobile device. This parameter can take of the following values:

SMSENC_ASCII	(0x0) The SMS buffer contains ASCII text and will be returned “as is” unmodified.
SMSENC_BINHEX	(0x1) Each byte in the SMS message buffer is represented with 2 ASCII hex characters. For example the buffer contains the following binary data: 0x30,0x31,0x46,0x46,0x45,0x42 or in other terms string “01FFEB” shall be decoded after reading as follows: {0x01,0xFF,0xEB}

#### *is\_encryption\_on*

- Set to 1 to decrypt the content of SMS buffer after reading. The text encoding shall be automatically forced to SMSENC\_BINHEX, regardless of the value of the *text\_encoding* parameter. Set to 0 to read SMS buffer without decryption.

#### *encryption\_alg*

- If *is\_encryption\_on* is set to 1, this parameter specifies the encryption algorithm to be used, one of the following:

- 0 – AES
- 1 – DES
- 2 – 3DES
- 3 – ARC4,
- 4 – CAST128

5 – BLOWFISH  
6 – TWOFISH

*encryption\_key*

- If *is\_encryption\_on* is set to 1, this parameter is the buffer of type `StringBuilder` containing encryption key.

*encrypt\_key\_bits*

- If *is\_encryption\_on* is set to 1, this parameter specifies the length of encryption key in number of bits

#### Return values:

HS_SMS_RC_OK	Success, the command to read SMS message is submitted. The actual result of read operation (an the SMS buffer read) should be obtained by polling with method <b>HsSmsDllReadEventsCs</b>
HS_SMS_RC_INV_PAR	Invalid parameters
HS_SMS_RC_NOT_INIT	HsSmsDll library not initialised
HS_SMS_RC_BAD_PORT	Invalid port number
HS_SMS_RC_PORT_NOT_OPEN	Specified port is not open
HS_SMS_RC_INV_STATE	Invalid internal state for reading SMS

#### Sample usage:

```
using HsSmsDllCs;  
  
StringBuilder key = new StringBuilder(textBoxKey.Text);  
  
rc = HsSms.HsSmsDllReadCs(cur_port,  
    index,  
    comboBoxEnc.SelectedIndex,  
    (checkBoxCrypt.Enabled)?1:0,  
    comboBoxAlg.SelectedIndex,  
    key,  
    key_bitsize  
);
```

### 2.1.5 HsSmsDllDeleteCs

#### Declaration:

```
[DllImport("HsSmsDll.dll")]  
public static extern int HsSmsDllDeleteCs(  
    int portno, // COM port number, zero based  
    int msg_index // index of message to delete  
);
```

#### Summary:

Use this function to delete an SMS message from GSM device, given the message index of the message in mobile device storage.

### Parameters:

*portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

*msg\_index*

– Message index of SMS message to delete in the mobile device storage.

### Return values:

HS_SMS_RC_OK	Success, the command to delete SMS message is submitted. The actual result of read operation should be obtained by polling with method <b>HsSmsDllReadEventsCs</b>
HS_SMS_RC_NOT_INIT	HsSmsDll library not initialised
HS_SMS_RC_BAD_PORT	Invalid port number
HS_SMS_RC_PORT_NOT_OPEN	Specified port is not open
HS_SMS_RC_INV_STATE	Invalid internal state for reading SMS

### Sample usage:

```
using HsSmsDllCs;
```

```
rc = HsSms.HsSmsDllDeleteCs(cur_port, index);
```

## 2.1.6 HsSmsDllReadEventsCs

### Declaration:

```
[DllImport("HsSmsDll.dll")]
public static extern int HsSmsDllReadEventsCs(
    int portno, // COM port number, zero based
    ref int evt, // event
    ref int arg1, // argument 1
    ref int arg2, // argument 2
    ref int arg3, // argument 3
    StringBuilder status, // new message status
    ref int status_len, // new message status length
    StringBuilder sender_number, // GSM sender number
    ref int number_len, // sender number length
    StringBuilder timestamp, // message timestamp
    ref int timestamp_len, // timestamp length
    StringBuilder msg_text, // message text
    ref int msg_text_len // message text length
);
```

### Summary:

This function should be called periodically from a timer, recommended every 10 ms to determine completion result of submitted commands and to detect the arrival of new SMS messages

**Parameters:**

*portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

*evt*

- If return code is HS\_SMS\_RC\_OK, this variable contains HsSmsDll event. This variable is passed by reference and filled by HsSmsDll with an event read from internal event queue. If no events have occurred the return value of the function is HS\_SMS\_RC\_NO\_EVENTS.

The possible events are listed below:

<b>Event</b>	<b>Description</b>
HS_SMSDLL_EV_OPEN_OK	Open operation (HsSmsDllOpenCs) completed with success
HS_SMSDLL_EV_SEND_OK	SMS send operation (HsSmsDllSendCs) completed with success, SMS message has been send by mobile device, Arg1 is GSM 03.40 TP-Message-Reference in integer format
HS_SMSDLL_EV_DELETE_OK	Delete operation (HsSmsDllDeleteCs) completed with success – SMS message with specified message storage index has been deleted. Arg1 is message index of the deleted message as specified in the call to HsSmsDllDeleteCs
HS_SMSDLL_EV_READ_OK	SMS read operation (HsSmsDllReadCs) completed with success, and:  status – contains the string indicating the status of message in mobile device memory, "REC UNREAD" or "REC READ"  status_len – length of status string  sender_number – GSM number of the sender of SMS message  number_len – length of sender number  timestamp – contains GSM 03.40 TP-Discharge-Time in time-string format: "yy/MM/dd, hh:mm:ss+zz", where characters indicate year (two last digits), month, day, hour, minutes, seconds and time zone. For example, 6th of May 1994, 22:10:00 GMT+2 hours equals "94/05/06,22:10:00+08"  timestamp_len – length of timestamp string  msg_text – SMS message text buffer. The buffer may contain plain text ASCII characters or non printable binary data depending on text encoding specified in call to HsSmsDllReadCs.  msg_text_len – length of SMS text buffer  Arg3 contains message read operation result code: HS_SMS_RC_OK – message is read correctly  HS_SMS_RC_ENCRYPT_ERR_ALG – decryption error, invalid

	<p>encryption algorithm</p> <p>HS_SMS_RC_ENCRYPT_ERR_INVP – decryption error, invalid parameter</p> <p>HS_SMS_RC_ENCRYPT_ERR_KEYSZ - decryption error, invalid key size</p> <p>HS_SMS_RC_ENCRYPT_ERR – decryption error, no additional information</p>
HS_SMSDLL_EV_NEW_RECEIVED	<p>Mobile device has received new SMS message. Arg1 contains SMS message index in the mobile device storage. The message may be read from mobile device using function HsSmsDIIReadCs</p>
HS_SMSDLL_EV_OPEN_FAIL	<p>Open operation (HsSmsDIIOpenCs) failed.</p> <p>Arg1 – error code, one of the following:</p> <ul style="list-style-type: none"> <li>0 – error waiting for response to AT command</li> <li>1 – GSM modem responded with error to AT command</li> <li>2 - +CMS error</li> <li>3 - +CME error</li> <li>4 – transmission to GSM modem failed, serial port error</li> <li>5 - supplied PIN number is incorrect format</li> </ul>
HS_SMSDLL_EV_SEND_FAIL	<p>Send operation (HsSmsDIISendCs) failed.</p> <p>Arg1 – error code, one of the following:</p> <ul style="list-style-type: none"> <li>0 – error waiting for response to AT command</li> <li>1 – GSM modem responded with error to AT command</li> <li>2 - +CMS error</li> <li>3 - +CME error</li> <li>4 – transmission to GSM modem failed, serial port error</li> </ul> <p>Arg2 – indicates if COM port has been closed or not:</p> <ul style="list-style-type: none"> <li>0 – SMS port is still open, new Send, Read or Delete operations possible</li> <li>1 – SMS port is closed, port must be open again before any other operations are possible</li> </ul>

<p>HS_SMSDLL_EV_DELETE_FAIL</p>	<p>Delete operation (HsSmsDIIDeleteCs) failed.  Arg1 – error code, one of the following:  0 – error waiting for response to AT command</p> <p>1 – GSM modem responded with error to AT command</p> <p>2 - +CMS error</p> <p>3 - +CME error</p> <p>4 – transmission to GSM modem failed, serial port error</p> <p>Arg2 – indicates if COM port has been closed or not:  0 – SMS port is still open, new Send, Read or Delete operations possible  1 – SMS port is closed, port must be open again before any other operations are possible</p>
<p>HS_SMSDLL_EV_READ_FAIL</p>	<p>Read operation (HsSmsDIIReadCs) failed.</p> <p>Arg1 – error code, one of the following:  0 – error waiting for response to AT command</p> <p>1 – GSM modem responded with error to AT command</p> <p>2 - +CMS error</p> <p>3 - +CME error</p> <p>4 – transmission to GSM modem failed, serial port error</p> <p>6 – error parsing received SMS message</p> <p>7 – no message found at specified message storage index</p> <p>Arg2 – indicates if COM port has been closed or not:  0 – SMS port is still open, new Send, Read or Delete operations possible  1 – SMS port is closed, port must be open again before any other operations are possible</p>
<p>HS_SMSDLL_EV_READ_SMSC_FAIL  <i>HsSmsDIIGetSmscCS failed</i></p>	<p>SMSC number (sms centre number) read command failed</p> <p>Arg1 – error code, one of the following:  0 – error waiting for response to AT command  1 – GSM modem responded with error to AT command  2 - +CMS error  3 - +CME error  4 – transmission to GSM modem failed, serial port error</p> <p>Arg2 – indicates if COM port has been closed or not:  0 – SMS port is still open, new Send, Read or Delete operations possible  1 – SMS port is closed, port must be open again before any other</p>

	operations are possible  Arg 3 – pointer to unsigned char string (null terminated) describing the detail of +CMS or +CME error. This parameter may be NULL
HS_SMSDLL_EV_SMSC_READ_OK  <i>HsSmsDllGetSmscCS completed with Success</i>	The SMSC number must now be retrieved with HsSmsDllReadSmsc function

*arg1*

- argument 1, the meaning of this parameter depends on the value of evt, see description of evt parameter

*arg2*

- argument 1, the meaning of this parameter depends on the value of evt, see description of evt parameter

*arg3*

- argument 1, the meaning of this parameter depends on the value of evt, see description of evt parameter

*status*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, status contains the string indicating the status of message in mobile device memory, "REC UNREAD" or "REC READ"

*status\_len*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, contains the length of status string

*sender\_number*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, contains GSM number of the sender of SMS message

*number\_len*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, contains the length of the sender number

*timestamp*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, contains GSM 03.40 TP-Discharge-Time in time-string format: "yy/MM/dd, hh:mm:ss+zz", where characters indicate year (two last digits), month, day, hour, minutes, seconds and time zone. For example, 6th of May 1994, 22:10:00 GMT+2 hours equals "94/05/06,22:10:00+08"

*timestamp\_len*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, contains the length of timestamp string

*msg\_text*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, contains SMS message text buffer. The buffer may contain plain text ASCII characters or non printable binary data depending on text encoding specified in call to HsSmsDllReadCs.

*msg\_text\_len*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, contains the length of SMS text buffer

### Return values:

HS_SMS_RC_NO_EVENTS	There are no events in the HsSmsDll event queue. All function parameters passed by reference must be ignored.
HS_SMS_RC_OK	An event has been read from HsSmsDll event queue, all function parameters passed by reference are valid

### Sample usage:

```
using HsSmsDllCs;
```

```
private void timer1_Tick(object sender, EventArgs e)
{
    int rc = 0;
    int evt = 0;
    int arg1 = 0;
    int arg2 = 0;
    int arg3 = 0;
    int portno;
    String errorString = "";

    int number_len = 0;
    int status_len = 0;
    int timestamp_len = 0;
    int text_len = 0;
    StringBuilder sender_number = new StringBuilder(31);
    StringBuilder stat = new StringBuilder(80);
    StringBuilder ts = new StringBuilder(80);
    StringBuilder msgtext = new StringBuilder(161);
    StringBuilder errstr = new StringBuilder(100);
    int errstr_len = 0;

    portno = cur_port;

    // read any queued event indications
    rc = HsSms.HsSmsDllReadEventsCs(portno, // port number
        ref evt, ref arg1, ref arg2, ref arg3, // event and 3 arguments
        stat, // status of message
        ref status_len, // length of status
        sender_number, // message sender number
        ref number_len, // number length
        ts, // timestamp
        ref timestamp_len, // timestamp length
        msgtext, // message text
        ref text_len); // length of text

    if (rc == HS_SMS_RC_NO_EVENTS) // return if no events queued
        return;

    switch (evt)
    {
        // SMS port open success
        case HS_SMSDLL_EV_OPEN_OK:
            errorString = System.String.Format(
                "Port ({0}) Open Okay: modem configured", portno);
    }
}
```

```

        status.Text = errorString;

        buttonsEnableOnPortOpen(sender, e);
        break;

// SMS port open failed
case HS_SMSDLL_EV_OPEN_FAIL:
    errorString = System.String.Format(
        "Port Closed (modem configuration failed) (event {0})", evt);
    status.Text = errorString;

    buttonsDisableOnPortClosed(sender, e);
    break;

// SMS send operation complete with failure
case HS_SMSDLL_EV_SEND_FAIL:
    errorString = System.String.Format("SMS Send Fail: ({0})",
        (arg2 == SMS_OPEN) ? "port still open" : "port closed");
    status.Text = errorString;

    actionsOnOpenOrClosed(sender, e, arg2);
    break;

// SMS send operation complete with success
case HS_SMSDLL_EV_SEND_OK:
    errorString = System.String.Format("SMS Sent OK: msg reference {0}",
        arg1);
    status.Text = errorString;
    buttonsEnableOnPortOpen(sender, e);
    break;

// New SMS received notification
case HS_SMSDLL_EV_NEW_RECEIVED:
    errorString = System.String.Format(
        "New SMS Received (index {0}), Click HsSmsDllRead to read", arg1);
    status.Text = errorString;

    errorString = System.String.Format("{0}", arg1);
    textBoxReadIndex.Text = errorString;
    textBoxDeleteIndex.Text = errorString;
    break;

// SMS read operation complete with success
case HS_SMSDLL_EV_READ_OK:
    status.Text = "Message read complete: Success";

    // first clear all fields
    labelStat.Text = "";
    textBoxNumber.Text = "";
    label_ts.Text = "";
    textBoxSms.Text = "";

    // then set all fields
    if (status_len != 0)
        labelStat.Text = stat.ToString();

    if (number_len != 0)

```

```

        textBoxNumber.Text = sender_number.ToString();

    if (timestamp_len != 0)
        label_ts.Text = ts.ToString();

    if (text_len != 0)
        textBoxSms.Text = msgtext.ToString();

    buttonRead.Enabled = true;
    break;

// SMS read operation complete with failure
case HS_SMSDLL_EV_READ_FAIL:
    HsSms.HsSmsDllGetErrCs(arg1, errstr, ref errstr_len);
    errorString = System.String.Format("SMS Read Fail: {0} ({1})",
    errstr, (arg2 == SMS_OPEN) ? "port still open" : "port closed");
    status.Text = errorString;

    actionsOnOpenOrClosed(sender, e, arg2);
    break;

// SMS delete operation complete with success
case HS_SMSDLL_EV_DELETE_OK:
    errorString = System.String.Format("SMS Delete OK: msg {0}", arg1);
    status.Text = errorString;
    buttonsEnableOnPortOpen(sender, e);
    textBoxDeleteIndex.Text = "";
    break
// SMS delete operation complete with failure
case HS_SMSDLL_EV_DELETE_FAIL:
    HsSms.HsSmsDllGetErrCs(arg1, errstr, ref errstr_len);
    errorString = System.String.Format("SMS Delete Fail: {0} ({1})",
    errstr, (arg2 == SMS_OPEN) ? "port still open" : "port closed");
    status.Text = errorString;

    actionsOnOpenOrClosed(sender, e, arg2);
    break;

// unrecognized event
default:
    errorString = System.String.Format(
    "unrecognized callback event {0}", evt);
    status.Text = errorString;
    break;
}
}

```

## 2.1.7 HsSmsDllGetErrCs

### Declaration:

```

[DllImport("HsSmsDll.dll")]
public static extern int HsSmsDllGetErrCs(
    int          error,          // error
    StringBuilder errstr,       // returned error string
    ref int      errstr_len     // length of error string
);

```

### **Summary:**

Use this function to get a descriptive string, corresponding to HsSmsDll error code

### **Parameters:**

*error*

- integer error code, returned in argument 1 (arg1) of HsSmsDllReadEventsCs function with events: HS\_SMSDLL\_EV\_OPEN\_FAIL, HS\_SMSDLL\_EV\_SEND\_FAIL, HS\_SMSDLL\_EV\_DELETE\_FAIL and HS\_SMSDLL\_EV\_READ\_FAIL. This error code provides additional information about the error. The error code is one of the following:

- 0 – error waiting for response to AT command
- 1 – GSM modem responded with error to AT command
- 2 - +CMS error
- 3 - +CME error
- 4 – transmission to GSM modem failed, serial port error
- 5 – Open failed, PIN supplied incorrect format
- 6 – error parsing received SMS message
- 7 – no message found at specified message storage index

*errstr*

- error string of StringBuilder type to receive the error string corresponding to integer error code, may be one of the following:

- "No Response To AT Command"
- "Error Response to AT Command"
- "+CMS Error"
- "+CME Error"
- "Incorrect PIN format"
- "Error parsing read message"
- "Message not found"

*errstr\_len*

- integer variable passed by reference which receives the number of characters in errstr.

### **Return values:**

HS_SMS_RC_OK	Success, errstr contains error string
HS_SMS_RC_INV_PAR	Error string corresponding to error code not found

### **Sample usage:**

```
using HsSmsDllCs;  
  
String errorString = "";  
StringBuilder errstr = new StringBuilder(100);  
int errstr_len = 0;  
  
HsSms.HsSmsDllGetErrCs(arg1, errstr, ref errstr_len);  
status.Text = errorString;
```

## 2.1.8 HsSmsDIIGetSmscCS

### Declaration:

```
[DllImport("HsSmsDII.dll")]
public static extern int HsSmsDIIGetSmscCS(
    int portno
);
```

### Summary:

Use this function to obtain SMSC number (short message service centre number) stored in GSM device. This command initiates (sends) the command to retrieve SMSC number. The result of command is reported asynchronously by reading events using HsSmsDIIReadCs function

### Parameters:

*portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

### Return values:

HS_SMS_RC_OK	Success, commad is submitted and is in progress
HS_SMS_RC_INV_PAR	Invalid parameters

## 2.1.9 HsSmsDIIReadSmsc

### Declaration:

```
[DllImport("HsSmsDII.dll")]
public static extern int HsSmsDIIReadSmsc(
    int portno,
    StringBuilder pdest
);
```

### Summary:

Use this function to read the SMSC number retrieved after successful completion of HsSmsDIIGetSmscCS function. This function may only be called after HsSmsDIIGetSmscCS completed with success and event HS\_SMSDLL\_EV\_SMSC\_READ\_OK has been received in HsSmsDIIReadCs function

### Parameters:

*portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

*pdest*

- String of type StringBuilder in C Sharp application where HsSmsDII shall copy the SMSC number string previously retrieved from the GSM device with HsSmsDIIGetSmscCS function

### Return values:

HS_SMS_RC_OK	Success, SMSC number is retrieved
HS_SMS_RC_INV_PAR	Invalid parameters

## 2.2 Visual C / C++ (v6 and 2003) Interface

### 2.2.1 HsSmsDllOpen

#### Declaration:

```
long __stdcall HsSmsDllOpen(  
    long          portno,           // COM port number, zero based  
    long          baudrate,        // baudrate constant  
    long          databits,       // databits constant  
    long          parity,          // parity constant  
    long          stopbits,       // stopbits constant  
    long          use_fc,          // use hardware flow control  
    long          callback_addr,   // function address in user code  
    unsigned char *pin,           // 4 ASCII digits SIM PIN, set NULL is not required  
    long          init_wait_secs,  // Initialization delay, 0 if no delay required  
    long          enable_log,     // 1=log event to file HsSms.log, 0=no logging  
    long          *rc              // return code pointer  
);
```

#### Summary:

Use this function to open and configure GSM device attached to specified serial port. This function must be called prior to calling other library functions.

#### Parameters:

##### *portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

##### *baudrate*

– Serial port rate in bauds, one of the following: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200, 128000, 256000

##### *databits*

– Number of data bits, one of the following: 5, 6, 7, 8

##### *parity*

– Parity code as follows: 2 - EVEN, 3 - MARK, 0 - NONE, 1 - ODD, 4 - SPACE

##### *stopbits*

– Stop bits code as follows: 0 – 1 Stop bit, 1 – 1.5 Stop bits, 2 – 2 Stop bits

##### *use\_fc*

– Set to 1 to use hardware RTS/CTS flow control

##### *callback\_addr*

– address of callback function in user code which is called by HsSmsDll to indicate results of completion of asynchronous operations and to inform of new SMS received. [See 2.2.7 “Callback Function”](#)

##### *\*pin*

– 4 digit SIM PIN in zero terminated ASCII string format. Set to NULL if your SIM does not require PIN.

##### *init\_wait\_secs*

– GSM hardware initialisation delay in seconds. This is the delay between asserting DTR on the serial port being opened and sending first AT configuration command to the GSM device. This delay may be

necessary to allow a GSM module to initialise, before it accepts AT commands. Delay value depends on GSM module vendor. If delay is not requires, set to 0.

*enable\_log*

- 1=log events to file HsSms.log. 0=no logging

*\*rc*

– Pointer to variable to receive function return code. See “Return values”.

**Return values:**

HS_SMS_RC_OK	Success, GSM device configuration is in progress, the actual result of operation is reported asynchronously via event callback
HS_SMS_RC_BAD_PORT	Invalid port number
HS_SMS_RC_INV_PAR	Invalid parameter
HS_SMS_RC_PORT_ALR_OPEN	Port is already open
HS_SMS_RC_NO_MEM	No free memory
HS_SMS_RC_COM_OPEN_FAIL	Failed to open COM port

**Sample usage:**

```
#include "HsSmsDll_if.h"
```

```
// Load SMS Dll library
```

```
hinstLib = LoadLibrary("HsSmsDll.dll");
```

```
if (!hinstLib)
```

```
{
```

```
    MessageBox(0, "HsSmsDll.dll not found!", 0, 0);
```

```
    return 0;
```

```
}
```

```
// Get all procedure address
```

```
SmsOpenFn = (HsSmsDllOpenFn)GetProcAddress(hinstLib, "HsSmsDllOpen");
```

```
if (!SmsOpenFn)
```

```
{
```

```
    MessageBox(0, "Failed to Get HsSmsDllOpen procedure address", 0, 0);
```

```
    FreeLibrary(hinstLib);
```

```
    return 0;
```

```
}
```

```
rc = SmsOpenFn(0,
```

```
    115200,
```

```
    8,
```

```
    0,
```

```
    0,
```

```
    1,
```

```
    (long)smsdll_callback, // function address in user code
```

```
    NULL, // 4 ASCII digits SIM PIN, NULL if not required
```

```
    1, // init delay in seconds
```

```
    &rc); // return code pointer
```

```
    // COM port number, zero based
```

```
    // baudrate constant
```

```
    // databits constant
```

```
    // parity constant
```

```
    // stopbits constant
```

```
    // use hardware flow control
```

**2.2.2 HsSmsDllClose**

**Declaration:**

```

long __stdcall HsSmsDIIClose(
    int portno // COM port number, zero based
)

```

**Summary:**

Use this function to close serial port connected to a GSM device and release communications session and resources associated with it.

**Parameters:**

*portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

**Return values:**

HS_SMS_RC_OK	Success. SMS port closed.
HS_SMS_RC_BAD_PORT	Invalid port number
HS_SMS_RC_PORT_NOT_OPEN	SMS port is not open

**Sample usage:**

```

#include "HsSmsDII_if.h"

HINSTANCE hinstLib = NULL;
HsSmsDIICloseFn SmsCloseFn;

// Load SMS DII library
hinstLib = LoadLibrary("HsSmsDII.dll");
if (!hinstLib)
{
    MessageBox(0, "HsSmsDII.dll not found!", 0, 0);
    return 0;
}

SmsCloseFn = (HsSmsDIICloseFn)GetProcAddress(hinstLib, "HsSmsDIIClose");
if (!SmsCloseFn)
{
    MessageBox(0, "Failed to Get HsSmsDIIClose procedure address", 0, 0);
    FreeLibrary(hinstLib);
    return 0;
}

rc = SmsCloseFn(0);

```

### 2.2.3 HsSmsDIISend

**Declaration (C, C++):**

```

long __stdcall HsSmsDIISend(
    long portno, // COM port number, zero based
    unsigned char *smsc_number, // SMS center number
    unsigned char *dest_number, // destination number (null terminated string)
    unsigned char *sms_text, // message text
    long sms_text_len, // message text length
    long text_encoding, // message encoding (plain ascii or binary as Ascii Hex)
)

```

```

    long          is_encryption_on,      // 1= encryption enabled, 0=encryption disabled
    long          encryption_alg,        // encryption algorithm
    unsigned char *encryption_key,      // encryption key
    long          encrypt_key_bits,     // encryption key size in bits
    long          *rc                    // return code pointer
);

```

**Summary:**

This function is used to send an SMS message (optionally encrypted) to a GSM destination.

**Parameters:**

*portno*

- Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

*\*smc\_number*

- SMS Service centre number via which the SMS message is to be sent, this number is different for each GSM network operator. The number is a zero terminated ASCII string

*\*dest\_number*

- Destination GSM number where the SMS message is to be sent. The number is a zero terminated ASCII string.

*\*sms\_text*

- SMS message buffer to be sent

*sms\_text\_len*

- length of SMS message buffer in bytes

*text\_encoding*

- This parameter determines how the content of sms\_text buffer is encoded before it is sent to GSM module. This parameter can take of the following values:

SMSENC_ASCII	(0x0) The buffer sms_text is sent "as is" unmodified.
SMSENC_BINHEX	(0x1) Each byte from sms_text buffer is sent as 2 ASCII hex characters. For example the sms_text buffer contains the following binary data {0x01,0xFF,0xEB}.  This buffer shall be encoded and sent as follows: "01FFEB" (string) or in hex representation 0x30,0x31,0x46,0x46,0x45,0x42  Using this method the maximum length of sms_text buffer must not exceed 80 bytes

*is\_encryption\_on*

- Set to 1 to encrypt the content of sms\_text buffer before sending. The text encoding shall be automatically forced to SMSENC\_BINHEX, regardless of the value of the *text\_encoding* parameter. The maximum length of source sms\_text buffer must not exceed 80 bytes. Set to 0 to send SMS text without encryption.

*encryption\_alg*

- If `is_encryption_on` is set to 1, this parameter specifies the encryption algorithm to be used, one of the following:

- 0 – AES
- 1 – DES
- 2 – 3DES
- 3 – ARC4,
- 4 – CAST128
- 5 – BLOWFISH
- 6 – TWOFISH

*\*encryption\_key*

- If `is_encryption_on` is set to 1, this parameter is pointer to buffer containing encryption key.

*encrypt\_key\_bits*

- If `is_encryption_on` is set to 1, this parameter specifies the length of encryption key in bits

*\*rc*

– Pointer to variable to receive function return code. See “Return values”.

### Return values:

HS_SMS_RC_OK	Success, SMS message is submitted for sending. HsSmsDll reports the actual send result to the application via callback function specified as a parameter with HsSmsDllOpen function.
HS_SMS_RC_NOT_INIT	HsSmsDll library not initialised
HS_SMS_RC_BAD_PORT	Invalid port number
HS_SMS_RC_PORT_NOT_OPEN	Specified port is not open
HS_SMS_RC_INV_STATE	Invalid internal state for sending SMS
HS_SMS_RC_INV_PAR	Invalid parameters

### Sample usage:

```
#include "HsSmsDll_if.h"
```

```
// Load SMS Dll library
hinstLib = LoadLibrary("HsSmsDll.dll");
if (!hinstLib)
{
    MessageBox(0, "HsSmsDll.dll not found!", 0, 0);
    return 0;
}

SmsSendFn = (HsSmsDllSendFn)GetProcAddress(hinstLib, "HsSmsDllSend");
if (!SmsSendFn)
{
    MessageBox(0, "Failed to Get HsSmsDllSend procedure address", 0, 0);
    FreeLibrary(hinstLib);
    return 0;
}
```

```

rc = SmsSendFn          (cur_port,      // COM port number, zero based
                        smsc,          // SMS Center number
                        dn,            // destination number (null terminated string)
                        txt,           // message text
                        text_len,      // message text length
                        sms_enc,       // message encoding
                        encryption_enabled,
                        encryption_algorithm,
                        key,
                        (long)(strlen(key) * 8),
                        &rc);         // return code pointer

```

## 2.2.4 HsSmsDIIRead

### Declaration:

```

long __stdcall HsSmsDIIRead(
    long          portno,          // COM port number, zero based
    long          msg_index,       // index of message to read
    long          text_encoding,   // message encoding (plain ascii or binary as Ascii Hex)
    long          is_encryption_on, // 1= encryption enabled, 0=encryption disabled
    long          encryption_alg,  // encryption algorithm
    unsigned char *encryption_key, // encryption key
    long          encrypt_key_bits, // encryption key size in bits
    long          *rc              // return code pointer
);

```

### Summary:

Use this function to read an already received SMS message from GSM device, given the message index of the message in mobile device storage.

### Parameters:

#### *portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

#### *msg\_index*

– Message index of SMS message to read in the mobile device storage.

#### *text\_encoding*

- This parameter determines how the content of sms buffer is decoded after it read from mobile device. This parameter can take of the following values:

SMSENC_ASCII	(0x0) The SMS buffer contains ASCII text and will be returned “as is” unmodified.
SMSENC_BINHEX	(0x1) Each byte in the SMS message buffer is represented with 2 ASCII hex characters. For example the buffer contains the following binary data: 0x30,0x31,0x46,0x46,0x45,0x42 or in other terms string “01FFEB” shall be decoded after reading as follows: {0x01,0xFF,0xEB}

#### *is\_encryption\_on*

- Set to 1 to decrypt the content of SMS buffer after reading. The text encoding shall be automatically forced to SMSENC\_BINHEX, regardless of the value of the *text\_encoding* parameter. Set to 0 to read SMS buffer without decryption.

*encryption\_alg*

- If *is\_encryption\_on* is set to 1, this parameter specifies the encryption algorithm to be used, one of the following:

- 0 – AES
- 1 – DES
- 2 – 3DES
- 3 – ARC4,
- 4 – CAST128
- 5 – BLOWFISH
- 6 – TWOFISH

*encryption\_key*

- If *is\_encryption\_on* is set to 1, this parameter is the buffer containing encryption key.

*encrypt\_key\_bits*

- If *is\_encryption\_on* is set to 1, this parameter specifies the length of encryption key in number of bits

*\*rc*

– Pointer to variable to receive function return code. See “Return values”.

**Return values:**

HS_SMS_RC_OK	Success, the command to read SMS message is submitted. The actual result of read operation (an the SMS buffer read) should be obtained while processing the corresponding events in the callback function, <a href="#">see 2.2.7 “Callback function”</a>
HS_SMS_RC_INV_PAR	Invalid parameters
HS_SMS_RC_NOT_INIT	HsSmsDII library not initialised
HS_SMS_RC_BAD_PORT	Invalid port number
HS_SMS_RC_PORT_NOT_OPEN	Specified port is not open
HS_SMS_RC_INV_STATE	Invalid internal state for reading SMS

**Sample usage:**

```
#include "HsSmsDII_if.h"
```

```
HINSTANCE  
HsSmsDIIReadFn
```

```
hinstLib = NULL;  
SmsReadFn;
```

```

// Load SMS DII library
hinstLib = LoadLibrary("HsSmsDII.dll");
if (!hinstLib)
{
    MessageBox(0, "HsSmsDII.dll not found!", 0, 0);
    return 0;
}

SmsReadFn = (HsSmsDIIReadFn)GetProcAddress(hinstLib, "HsSmsDIIRead");
if (!SmsReadFn)
{
    MessageBox(0, "Failed to Get HsSmsDIIRead procedure address", 0, 0);
    FreeLibrary(hinstLib);
    return 0;
}

```

```

    }

    rc = SmsReadFn(cur_port, msg_index, sms_enc, encryption_enabled, encryption_algorithm, key,
                  (long)(strlen(key) * 8), &rc);

```

## 2.2.5 HsSmsDIIDelete

### Declaration:

```

long __stdcall HsSmsDIIDelete(
    long          portno,           // COM port number, zero based
    long          msg_index,       // index of message to delete
    long          *rc              // return code pointer
);

```

### Summary:

Use this function to delete an SMS message from GSM device, given the message index of the message in mobile device storage.

### Parameters:

*portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

*msg\_index*

– Message index of SMS message to delete in the mobile device storage.

*\*rc*

– Pointer to variable to receive function return code. See “Return values”.

### Return values:

HS_SMS_RC_OK	Success, the command to delete SMS message is submitted. The actual result of delete operation should be obtained while processing the corresponding events in the callback function, <a href="#">see 2.2.7 “Callback function”</a>
HS_SMS_RC_NOT_INIT	HsSmsDII library not initialised
HS_SMS_RC_BAD_PORT	Invalid port number
HS_SMS_RC_PORT_NOT_OPEN	Specified port is not open
HS_SMS_RC_INV_STATE	Invalid internal state for reading SMS

### Sample usage:

```
#include "HsSmsDII_if.h"
```

```

HINSTANCE          hinstLib = NULL;
HsSmsDIIDeleteFn  SmsDeleteFn;

```

```

SmsDeleteFn = (HsSmsDllDeleteFn)GetProcAddress(hinstLib, "HsSmsDllDelete");
if (!SmsDeleteFn)
{
    MessageBox(0, "Failed to Get HsSmsDllDelete procedure address", 0, 0);
    FreeLibrary(hinstLib);
    return 0;
}

rc = SmsDeleteFn(cur_port, msg_index, &rc);

```

## 2.2.6 HsSmsDllGetSmSC

### Declaration:

```

long __stdcall HsSmsDllGetSmSC(
    long portno, // COM port number, zero based
    long *rc // HS SMS return code
)

```

### Summary:

Use this function to obtain SMSC number (short message service centre number) stored in GSM device. This command initiates (sends) the command to retrieve SMSC number. The result of command is reported asynchronously via event callback

### Parameters:

*portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

*\*rc*

– Pointer to variable to receive function return code. See “Return values”.

### Return values:

HS_SMS_RC_OK	Success, command is submitted and is in progress
HS_SMS_RC_INV_PAR	Invalid parameters

## 2.2.7 HsSmsDllExErrorStr

### Declaration:

```

long __stdcall HsSmsDllExErrorStr(long ex_error, long copy_addr, long *str_len, long *rc);

```

### Summary:

Use this function to get a descriptive string, corresponding to HsSmsDll error code

### Parameters:

*error*

– error code, returned in argument 1 (arg1) of [callback function](#) with events:

HS\_SMSDLL\_EV\_OPEN\_FAIL, HS\_SMSDLL\_EV\_SEND\_FAIL, HS\_SMSDLL\_EV\_DELETE\_FAIL and HS\_SMSDLL\_EV\_READ\_FAIL. This error code provides additional information about the error. The error code is one of the following:

0 – error waiting for response to AT command

- 1 – GSM modem responded with error to AT command
- 2 - +CMS error
- 3 - +CME error
- 4 – transmission to GSM modem failed, serial port error
- 5 – Open failed, PIN supplied incorrect format
- 6 – error parsing received SMS message
- 7 – no message found at specified message storage index

*copy\_addr*

- address of buffer in user application space to copy the error string into

*str\_len*

- pointer to long variable to receive the length of the error string

*\*rc*

- Pointer to variable to receive function return code. See “Return values”.

### Return values:

HS_SMS_RC_OK	Success, error string copied to copy_addr address
HS_SMS_RC_INV_PAR	Invalid parameters

### Sample usage:

```
#include "HsSmsDll_if.h"
```

```
HINSTANCE hinstLib = NULL;
```

```
HsSmsDllExErrorStrFn SmsExErrStrFn;
```

```
SmsExErrStrFn = (HsSmsDllExErrorStrFn)GetProcAddress(hinstLib, "HsSmsDllExErrorStr");
```

```
if (!SmsExErrStrFn)
```

```
{
```

```
    MessageBox(0, "Failed to Get HsSmsDllExErrorStr procedure address", 0, 0);
```

```
    FreeLibrary(hinstLib);
```

```
    return 0;
```

```
}
```

```
rc = SmsExErrStrFn(arg1, (long)str, &llen, &rc);
```

## 2.2.8 Callback Function

### Declaration:

```
long __stdcall c_callback(long portno, long event, long arg1, long arg2, long arg3);
```

### Summary:

When the user application opens SMS port with function HsSmsDllOpen, the user specifies a function callback pointer as one of parameters. This function callback receives asynchronous notifications of completion of HsSmsDll operations and notifications about the arrival of new SMS messages.

### Parameters:

*portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

event and arguments (arg1, arg2, arg3)

Event and Description	Arguments
<p>HS_SMSDLL_EV_OPEN_OK</p> <p><i>SMS read operation (HsSmsDllRead) completed with success</i></p>	<p>Arg1 – 0 Arg2 – 0 Arg3 – 0</p>
<p>HS_SMSDLL_EV_SEND_OK</p> <p><i>SMS Send operation (HsSmsDllSend) completed with success (message sent)</i></p>	<p>Arg1– message reference of the message sent Arg2 –0 Arg3 – 0</p>
<p>HS_SMSDLL_EV_READ_OK</p> <p><i>SMS Read operation (HsSmsDllRead) completed with success</i></p>	<p>Arg1 – pointer message read structure, defined as follows:</p> <pre>#define MAX_SMS_NUMBER      30      // max space required for GSM number #define MAX_SMS_TS          40      // max timestamp length in octets #define MAX_SMS_STATUS      40      // message status #define MAX_SMS_SEND_LEN    160     // max size of SMS text buffer  /* received message structure */ typedef struct {     unsigned char    status[MAX_SMS_STATUS+1]; // status     long             status_len;               // length of status      unsigned char    number[MAX_SMS_NUMBER+1]; // phone number     long             number_len;               // number length      unsigned char    timestamp[MAX_SMS_TS+1]; // timestamp     long             ts_len;                   // timestamp length      unsigned char    text[(MAX_SMS_SEND_LEN * 2)+1]; // message text     long             text_len;                 // text length } hs_sms_msg_t;</pre> <p>Notes: Status - contains the string indicating the status of message in mobile device memory, "REC UNREAD" or "REC READ"</p> <p>number – GSM number of the sender of SMS message</p> <p>timestamp - GSM 03.40 TP-Discharge-Time in time-string format: "yy/MM/dd,hh:mm:ss+zz", where characters indicate year (two last digits), month, day, hour, minutes, seconds and time zone. For example, 6th of May 1994, 22:10:00 GMT+2 hours equals "94/05/06,22:10:00+08"</p> <p>text – SMS message text buffer. The buffer may contain plain text ASCII characters or non printable binary data depending on text encoding specified in call to HsSmsDllRead</p> <p>Arg2 – reserved</p>

	Arg3 – 0
HS_SMSDLL_EV_DELETE_OK  <i>SMS Delete operation (HsSmsDllDelete) completed with success (message deleted)</i>	Arg1 – index of the message storage in the mobile device of the deleted message  Arg2 – 0 Arg3 – 0
HS_SMSDLL_EV_OPEN_FAIL  <i>HsSmsDllOpen failed</i>	Arg1 – error code, one of the following:  0 – error waiting for response to AT command 1 – GSM modem responded with error to AT command 2 - +CMS error 3 - +CME error 4 – transmission to GSM modem failed, serial port error 5 - supplied PIN number is incorrect format  Arg2 – 0  Arg 3 – pointer to unsigned char string (null terminated) describing the detail of +CMS or +CME error. This parameter may be NULL
HS_SMSDLL_EV_SEND_FAIL  <i>HsSmsDllSend failed</i>	Arg1 - error code, one of the following:  0 – error waiting for response to AT command 1 – GSM modem responded with error to AT command 2 - +CMS error 3 - +CME error 4 – transmission to GSM modem failed, serial port error  Arg2 – indicates if COM port has been closed or not: 0 – SMS port is still open, new Send, Read or Delete operations possible 1 – SMS port is closed, port must be open again before any other operations are possible  Arg 3 – pointer to unsigned char string (null terminated) describing the detail of +CMS or +CME error. This parameter may be NULL
HS_SMSDLL_EV_DELETE_FAIL  <i>HsSmsDllDelete failed</i>	Arg1 – error code, one of the following:  0 – error waiting for response to AT command 1 – GSM modem responded with error to AT command 2 - +CMS error 3 - +CME error 4 – transmission to GSM modem failed, serial port error  Arg2 – indicates if COM port has been closed or not: 0 – SMS port is still open, new Send, Read or Delete operations possible 1 – SMS port is closed, port must be open again before any other operations are possible  Arg 3 – pointer to unsigned char string (null terminated) describing

	the detail of +CMS or +CME error. This parameter may be NULL
<p>HS_SMSDLL_EV_READ_FAIL</p> <p><i>HsSmsDllRead failed</i></p>	<p>Arg1 – error code, one of the following:</p> <ul style="list-style-type: none"> <li>0 – error waiting for response to AT command</li> <li>1 – GSM modem responded with error to AT command</li> <li>2 - +CMS error</li> <li>3 - +CME error</li> <li>4 – transmission to GSM modem failed, serial port error</li> <li>6 – error parsing received SMS message</li> <li>7 – no message found at specified message storage index</li> </ul> <p>Arg2 – indicates if COM port has been closed or not:</p> <ul style="list-style-type: none"> <li>0 – SMS port is still open, new Send, Read or Delete operations possible</li> <li>1 – SMS port is closed, port must be open again before any other operations are possible</li> </ul> <p>Arg 3 – pointer to unsigned char string (null terminated) describing the detail of +CMS or +CME error. This parameter may be NULL</p>
<p>HS_SMSDLL_EV_NEW_RECEIVED</p> <p><i>Mobile device has received new SMS message.</i></p>	<p>Arg1 – SMS message index in the mobile device storage. The message may be read from mobile device using function HsSmsDllRead</p> <p>Arg2 – 0</p> <p>Arg3 – 0</p>
<p>HS_SMSDLL_EV_READ_SMSC_FAIL</p> <p><i>HsSmsDllGetSmsc failed</i></p>	<p>SMSC number (sms centre number) read command failed</p> <p>Arg1 – error code, one of the following:</p> <ul style="list-style-type: none"> <li>0 – error waiting for response to AT command</li> <li>1 – GSM modem responded with error to AT command</li> <li>2 - +CMS error</li> <li>3 - +CME error</li> <li>4 – transmission to GSM modem failed, serial port error</li> </ul> <p>Arg2 – indicates if COM port has been closed or not:</p> <ul style="list-style-type: none"> <li>0 – SMS port is still open, new Send, Read or Delete operations possible</li> <li>1 – SMS port is closed, port must be open again before any other operations are possible</li> </ul> <p>Arg 3 – pointer to unsigned char string (null terminated) describing the detail of +CMS or +CME error. This parameter may be NULL</p>
<p>HS_SMSDLL_EV_SMSC_READ_OK</p> <p><i>HsSmsDllGetSmsc completed with Success</i></p>	<p>Arg1 – pointer to SMSC number retrieved from GSM device</p> <p>Arg2 = 0</p> <p>Arg3 = 0</p>

### Return values:

Not used by HsSmsDll, return 0

### Sample usage:

```
/* callback from HsSmsDll.dll */
long __stdcall smsdll_callback(long portno, long event, long arg1, long arg2, long arg3)
{
    unsigned char s[200] = {0};

    switch (event)
    {
        // SMS Port open complete: Success. (Initial GSM modem configuration complete)
        case HS_SMSDLL_EV_OPEN_OK:
            on_sms_port_open_complete(portno, arg1, arg2, arg3);
            break;

        // Message sent successfully
        case HS_SMSDLL_EV_SEND_OK:
            on_sms_message_sent_ok(portno, arg1, arg2, arg3);
            break;

        // Message read complete
        case HS_SMSDLL_EV_READ_OK:
            on_sms_read_completed_ok(portno, arg1, arg2, arg3);
            break;

        // Message deleted successfully
        case HS_SMSDLL_EV_DELETE_OK:
            on_sms_message_delete_ok(portno, arg1, arg2, arg3);
            break;

        // SMS port open competed with failure.
        case HS_SMSDLL_EV_OPEN_FAIL:
            on_sms_port_open_fail(portno, arg1, arg2, arg3);
            break;

        // SMS sending competed with failure.
        case HS_SMSDLL_EV_SEND_FAIL:
            on_sms_send_fail(portno, arg1, arg2, arg3);
            break;

        // SMS delete completed with failure
        case HS_SMSDLL_EV_DELETE_FAIL:
            on_sms_delete_fail(portno, arg1, arg2, arg3);
            break;

        // SMS read completed with failure
        case HS_SMSDLL_EV_READ_FAIL:
            on_sms_read_fail(portno, arg1, arg2, arg3);
            break;

        // New SMS received notification
        case HS_SMSDLL_EV_NEW_RECEIVED:
            on_sms_new_received(portno, arg1, arg2, arg3);
    }
}
```

```

        break;
    }

    return 0;
}

```

## 2.3 Visual Basic (v6 and 2003) Interface

### 2.3.1 HsSmsDIIOpenVB

#### Declaration:

```

Declare Function HsSmsDIIOpenVB Lib "HsSmsDII.dll" _
    (ByVal portno As Long, _
    ByVal baudrate As Long, _
    ByVal databits As Long, _
    ByVal parity As Long, _
    ByVal stopbits As Long, _
    ByVal use_fc As Long, _
    ByVal pin As String, _
    ByVal init_wait_secs As Long, _
    ByVal enable_log As Long, _
    ByVal rc As Long) As Long

```

#### Summary:

Use this function to open and configure GSM device attached to specified serial port. This function must be called prior to calling other library functions.

#### Parameters:

##### *portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

##### *baudrate*

– Serial port rate in bauds, one of the following: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200, 128000, 256000

##### *databits*

– Number of data bits, one of the following: 5, 6, 7, 8

##### *parity*

– Parity code as follows: 2 - EVEN, 3 - MARK, 0 - NONE, 1 - ODD, 4 - SPACE

##### *stopbits*

– Stop bits code as follows: 0 – 1 Stop bit, 1 – 1.5 Stop bits, 2 – 2 Stop bits

##### *use\_fc*

– Set to 1 to use hardware RTS/CTS flow control

##### *pin*

– 4 digit SIM PIN in zero terminated ASCII string format. Set to NULL if your SIM does not require PIN.

*init\_wait\_secs*

- GSM hardware initialisation delay in seconds. This is the delay between asserting DTR on the serial port being opened and sending first AT configuration command to the GSM device. This delay may be necessary to allow a GSM module to initialise, before it accepts AT commands. Delay value depends on GSM module vendor. If delay is not required, set to 0.

*enable\_log*

- 1=log events to file HsSms.log. 0=no logging

*\*rc*

- Pointer to variable to receive function return code. See "Return values".

**Return values:**

HS_SMS_RC_OK	Success, GSM device configuration is in progress, the actual result of operation determined by polling with function HsSmsDIIReadEventsVB
HS_SMS_RC_BAD_PORT	Invalid port number
HS_SMS_RC_INV_PAR	Invalid parameter
HS_SMS_RC_PORT_ALR_OPEN	Port is already open
HS_SMS_RC_NO_MEM	No free memory
HS_SMS_RC_COM_OPEN_FAIL	Failed to open COM port

**Sample usage:**

Dim rc As Long

HsSmsDIIOpenVB 0, 115200, 8, 0, 0, 1, "", 3, ByVal VarPtr(rc)

If rc = HS\_SMS\_RC\_OK Then

    Form1.Status.Caption = "Open SMS port OK. Configuring ..."

Else

    Form1.Status.Caption = "SMS port open failed rc " & rc

End If

**2.3.2 HsSmsDIICloseVB**

**Declaration:**

Declare Function HsSmsDIICloseVB Lib "HsSmsDII.dll" \_  
(ByVal portno As Long, ByVal rc As Long) As Long

**Summary:**

Use this function to close an open SMS port. This function releases all resources associated with the port and closes the specified COM port

**Parameters:**

*portno*

- Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

*\*rc*

- Pointer to variable to receive function return code. See "Return values".

**Return values:**

HS_SMS_RC_OK	Success, SMS port closed
HS_SMS_RC_BAD_PORT	Invalid port number
HS_SMS_RC_INV_PAR	Invalid parameter
HS_SMS_RC_PORT_NOT_OPEN	Port is not open

**Sample usage:**

```
Dim rc As Long

HsSmsDIICloseVB 0, ByVal VarPtr(rc)
If rc = HS_SMS_RC_OK Then
    Form1.Status.Caption = "SMS port closed Ok"
Else
    Form1.Status.Caption = "SMS port close failed"
End If
```

**2.3.3 HsSmsDIISendVB**

**Declaration:**

```
Declare Function HsSmsDIISendVB Lib "HsSmsDII.dll" _
(ByVal portno As Long, _
ByVal smsc_number As String, _
ByVal dest_number As String, _
ByVal lpSmsText As Long, _
ByVal sms_text_len As Long, _
ByVal text_encoding As Long, _
ByVal is_encryption_on As Long, _
ByVal encryption_alg As Long, _
ByVal lpEncryptionKey As Long, _
ByVal encrypt_key_bits As Long, _
ByRef rc As Long) As Long
```

**Summary:**

Use this function to send SMS message. The port must be previously open with function HsSmsDIIOpenVB

**Parameters:**

*portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

*smsc\_number*

– SMS Service Centre number via which the SMS message is to be sent, this number is different for each GSM network operator. The number is of String type

*dest\_number*

– Destination GSM number where the SMS message is to be sent. The number is of String type

*lpSmsText*

– Address of SMS message buffer to send. The HsSmsDII.dll receives the buffer as ANSI byte array. Depending on selected encoding (see parameter *text\_encoding*) the content of the buffer may be ASCII printable characters or non ASCII binary content.

*sms\_text\_len*

- length of SMS message buffer in bytes.

*text\_encoding*

- This parameter determines how the content of sms\_text buffer is encoded before it is sent to GSM module. This parameter can take of the following values:

SMSENC_ASCII	(0x0) The buffer sms_text is sent "as is" unmodified.
SMSENC_BINHEX	(0x1) Each byte from sms_text buffer is sent as 2 ASCII hex characters. For example the sms_text buffer contains the following binary data {0x01,0xFF,0xEB}. This buffer shall be encoded and sent as follows: "01FFEB" (string) or in hex representation 0x30,0x31,0x46,0x46,0x45,0x42  Using this method the maximum length of sms_text buffer must not exceed 80 bytes

*is\_encryption\_on*

- Set to 1 to encrypt the content of sms\_text buffer before sending. The text encoding shall be automatically forced to SMSENC\_BINHEX, regardless of the value of the *text\_encoding* parameter. The maximum length of source sms\_text buffer must not exceed 80 bytes. Set to 0 to send SMS text without encryption.

*encryption\_alg*

- If is\_encryption\_on is set to 1, this parameter specifies the encryption algorithm to be used, one of the following:

- 0 – AES
- 1 – DES
- 2 – 3DES
- 3 – ARC4,
- 4 – CAST128
- 5 – BLOWFISH
- 6 – TWOFISH

*lpEncryptionKey*

- If is\_encryption\_on is set to 1, this parameter is the address of buffer containing encryption key (must be ANSI byte array)

*encrypt\_key\_bits*

- If is\_encryption\_on is set to 1, this parameter specifies the length of encryption key in number of bits

\*rc

– Pointer to variable to receive function return code. See "Return values".

### **Return values:**

HS_SMS_RC_OK	Success, SMS message is submitted for sending. The actual send result is determined by polling with method <b>HsSmsDllReadEventsVB</b>
HS_SMS_RC_NOT_INIT	HsSmsDll library not initialised
HS_SMS_RC_BAD_PORT	Invalid port number
HS_SMS_RC_PORT_NOT_OPEN	Specified port is not open
HS_SMS_RC_INV_STATE	Invalid internal state for sending SMS
HS_SMS_RC_INV_PAR	Invalid parameters

### Sample usage:

```
'Send SMS
Sub OnSend()
    Dim bAnsiSms() As Byte
    Dim bAnsiKey() As Byte
    Dim encrypt As Long
    Dim smslen As Long
    Dim keysize As Long
    Dim rc As Long

    encrypt = Form1.CheckEncryption.Value

    If Len(Form1.Smstext.Text) = 0 Then
        Form1.Status.Caption = "SMS text length cannot be 0 characters"
        Exit Sub
    End If

    bAnsiSms = StrConv(Form1.Smstext.Text, vbFromUnicode)
    smslen = Len(Form1.Smstext)

    keysize = 0
    bAnsiKey() = StrConv("0", vbFromUnicode)

    If encrypt = 1 Then
        If Len(Form1.TextKey.Text) = 0 Then
            Form1.Status.Caption = "Send Error - Encryption Key missing"
            Exit Sub
        End If

        bAnsiKey = StrConv(Form1.TextKey.Text, vbFromUnicode)
        keysize = (Len(Form1.TextKey.Text) * 8)
    End If

    HsSmsDIISendVB Form1.ComboCom.ListIndex, Form1.TextSmscNumber.Text, _
        Form1.TextGsmNumber.Text, VarPtr(bAnsiSms(0)), smslen, Form1.ComboEnc.ListIndex, _
        encrypt, Form1.ComboAlg.ListIndex, VarPtr(bAnsiKey(0)), keysize, ByVal VarPtr(rc)

    If rc = HS_SMS_RC_OK Then
        Form1.CommandSend.Enabled = False
        Form1.CommandRead.Enabled = False
        Form1.CommandDelete.Enabled = False
        Form1.Status.Caption = "Submitted SMS for sending OK. Sending in progress..."
    Else
        Form1.Status.Caption = "Submitted SMS for sending Fail rc = " & rc
    End If

End Sub
```

### **2.3.4 HsSmsDIIReadVB**

#### Declaration:

```
Declare Function HsSmsDIIReadVB Lib "HsSmsDII.dll" _
```

(ByVal portno As Long, \_  
 ByVal msg\_index As Long, \_  
 ByVal text\_encoding As Long, \_  
 ByVal is\_encryption\_on As Long, \_  
 ByVal encryption\_alg As Long, \_  
 ByVal lpEncryptionKey As Long, \_  
 ByVal encrypt\_key\_bits As Long, \_  
 ByVal rc As Long) As Long

**Summary:**

Use this function to read an already received SMS message from GSM device, given the message index of the message in mobile device storage.

**Parameters:**

*portno*

- Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

*msg\_index*

- Message index of SMS message to read in the mobile device storage.

*text\_encoding*

- This parameter determines how the content of sms buffer is decoded after it read from mobile device. This parameter can take of the following values:

SMSENC_ASCII	(0x0) The SMS buffer contains ASCII text and will be returned "as is" unmodified.
SMSENC_BINHEX	(0x1) Each byte in the SMS message buffer is represented with 2 ASCII hex characters. For example the buffer contains the following binary data: 0x30,0x31,0x46,0x46,0x45,0x42 or in other terms string "01FFEB" shall be decoded after reading as follows: {0x01,0xFF,0xEB}

*is\_encryption\_on*

- Set to 1 to decrypt the content of SMS buffer after reading. The text encoding shall be automatically forced to SMSENC\_BINHEX, regardless of the value of the *text\_encoding* parameter. Set to 0 to read SMS buffer without decryption.

*encryption\_alg*

- If *is\_encryption\_on* is set to 1, this parameter specifies the encryption algorithm to be used, one of the following:

- 0 – AES
- 1 – DES
- 2 – 3DES
- 3 – ARC4,
- 4 – CAST128
- 5 – BLOWFISH
- 6 – TWOFISH

*lpEncryptionKey*

- If *is\_encryption\_on* is set to 1, this parameter is the address of buffer containing encryption key (must by ANSI byte array)

*encrypt\_key\_bits*

- If *is\_encryption\_on* is set to 1, this parameter specifies the length of encryption key in number of bits

*\*rc*

- Pointer to variable to receive function return code. See "Return values".

**Return values:**

HS_SMS_RC_OK	Success, the command to read SMS message is submitted. The actual result of read operation (an the SMS buffer read) should be obtained by polling with method <b>HsSmsDllReadEventsVB</b>
HS_SMS_RC_INV_PAR	Invalid parameters
HS_SMS_RC_NOT_INIT	HsSmsDll library not initialised
HS_SMS_RC_BAD_PORT	Invalid port number
HS_SMS_RC_PORT_NOT_OPEN	Specified port is not open
HS_SMS_RC_INV_STATE	Invalid internal state for reading SMS

**Sample usage:**

```
Sub OnRead()  
    Dim bAnsiKey() As Byte  
    Dim encrypt As Long  
    Dim msg_index As Long  
    Dim keysize As Long  
    Dim rc As Long  
  
    encrypt = Form1.CheckEncryption.Value  
  
    keysize = 0  
    bAnsiKey() = StrConv("0", vbFromUnicode)  
  
    If encrypt = 1 Then  
        If Len(Form1.TextKey.Text) = 0 Then  
            Form1.Status.Caption = "Read Error - Encryption Key missing"  
            Exit Sub  
        End If  
  
        bAnsiKey = StrConv(Form1.TextKey.Text, vbFromUnicode)  
        keysize = (Len(Form1.TextKey.Text) * 8)  
    End If  
  
    msg_index = Val(Form1.TextRindex.Text)  
  
    HsSmsDllReadVB Form1.ComboCom.ListIndex, msg_index, Form1.ComboEnc.ListIndex, encrypt,  
    Form1.ComboAlg.ListIndex, _  
    VarPtr(bAnsiKey(0)), keysize, ByVal VarPtr(rc)  
  
    If rc = HS_SMS_RC_OK Then  
        Form1.CommandSend.Enabled = False  
        Form1.CommandRead.Enabled = False  
        Form1.CommandDelete.Enabled = False  
        Form1.Status.Caption = "Submitted Read SMS command OK. Read in progress..."  
    Else  
        Form1.Status.Caption = "Submitted Read SMS command Fail rc = " & rc  
    End If  
End Sub
```

End If

End Sub

### 2.3.5 HsSmsDIIDelete

#### Declaration:

```
Declare Function HsSmsDIIDelete Lib "HsSmsDII.dll" _  
(ByVal portno As Long, _  
ByVal msg_index As Long, _  
ByRef rc As Long) As Long
```

#### Summary:

Use this function to delete an SMS message from GSM device, given the message index of the message in mobile device storage.

#### Parameters:

*portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

*msg\_index*

– Message index of SMS message to delete in the mobile device storage.

*\*rc*

– Pointer to variable to receive function return code. See “Return values”.

#### Return values:

HS_SMS_RC_OK	Success, the command to delete SMS message is submitted. The actual result of delete operation should be obtained by polling with function <b>HsSmsDIIReadEventsVB</b>
HS_SMS_RC_NOT_INIT	HsSmsDII library not initialised
HS_SMS_RC_BAD_PORT	Invalid port number
HS_SMS_RC_PORT_NOT_OPEN	Specified port is not open
HS_SMS_RC_INV_STATE	Invalid internal state for reading SMS

#### Sample usage:

```
Sub OnDelete()
```

```
Dim msg_index As Long
```

```
Dim rc As Long
```

```
msg_index = Val(Form1.TextDIndex.Text)
```

```
HsSmsDIIDelete Form1.ComboCom.ListIndex, msg_index, ByVal VarPtr(rc)
```

```

If rc = HS_SMS_RC_OK Then
    Form1.CommandSend.Enabled = False
    Form1.CommandRead.Enabled = False
    Form1.CommandDelete.Enabled = False
    Form1.Status.Caption = "Submitted Delete SMS command OK. Delete in progress..."
Else
    Form1.Status.Caption = "Submitted Delete SMS command Fail rc = " & rc
End If
End Sub

```

### 2.3.6 HsSmsDIIReadEventsVB

#### Declaration:

```

Declare Function HsSmsDIIReadEventsVB Lib "HsSmsDII.dll" _
(ByVal portno As Long, _
ByRef evt As Long, _
ByRef arg1 As Long, _
ByRef arg2 As Long, _
ByRef arg3 As Long, _
ByVal IpStatus As Long, _
ByRef status_len As Long, _
ByVal IpSender As Long, _
ByRef number_len As Long, _
ByVal IpTimestamp As Long, _
ByRef timestamp_len As Long, _
ByVal IpText As Long, _
ByRef msg_text_len As Long, _
ByRef rc As Long _
) As Long

```

#### Summary:

This function should be called periodically from a timer, recommended every 10 ms to determine completion result of submitted commands and to detect the arrival of new SMS messages

#### Parameters:

##### *portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

##### *evt*

- If return code is HS\_SMS\_RC\_OK, this variable contains HsSmsDII event. This variable is passed by reference and filled by HsSmsDII with an event read from internal event queue. If no events have occurred the return value of the function is HS\_SMS\_RC\_NO\_EVENTS.

The possible events are listed below:

Event	Description
HS_SMSDLL_EV_OPEN_OK	Open operation (HsSmsDIIOpenCs) completed with success

HS_SMSDLL_EV_SEND_OK	SMS send operation (HsSmsDIISendCs) completed with success, SMS message has been send by mobile device, Arg1 is GSM 03.40 TP-Message-Reference in integer format
HS_SMSDLL_EV_DELETE_OK	Delete operation (HsSmsDIIDeleteCs) completed with success – SMS message with specified message storage index has been deleted. Arg1 is message index of the deleted message as specified in the call to HsSmsDIIDeleteCs
HS_SMSDLL_EV_READ_OK	<p>SMS read operation (HsSmsDIIReadCs) completed with success, and:</p> <p>status – contains the string indicating the status of message in mobile device memory, "REC UNREAD" or "REC READ"</p> <p>status_len – length of status string</p> <p>sender_number – GSM number of the sender of SMS message</p> <p>number_len – length of sender number</p> <p>timestamp – contains GSM 03.40 TP-Discharge-Time in time-string format: "yy/MM/dd, hh:mm:ss+zz", where characters indicate year (two last digits), month, day, hour, minutes, seconds and time zone. For example, 6th of May 1994, 22:10:00 GMT+2 hours equals "94/05/06,22:10:00+08"</p> <p>timestamp_len – length of timestamp string</p> <p>msg_text – SMS message text buffer. The buffer may contain plain text ASCII characters or non printable binary data depending on text encoding specified in call to HsSmsDIIReadCs.</p> <p>msg_text_len – length of SMS text buffer</p> <p>Arg3 contains message read operation result code:  HS_SMS_RC_OK – message is read correctly</p> <p>HS_SMS_RC_ENCRYPT_ERR_ALG – decryption error, invalid encryption algorithm</p> <p>HS_SMS_RC_ENCRYPT_ERR_INVP – decryption error, invalid parameter</p> <p>HS_SMS_RC_ENCRYPT_ERR_KEYSZ - decryption error, invalid key size</p> <p>HS_SMS_RC_ENCRYPT_ERR – decryption error, no additional information</p>
HS_SMSDLL_EV_NEW_RECEIVED	Mobile device has received new SMS message. Arg1 contains SMS message index in the mobile device storage. The message may be read from mobile device using function HsSmsDIIReadCs
HS_SMSDLL_EV_OPEN_FAIL	Open operation (HsSmsDIIOpenCs) failed.

	<p>Arg1 – error code, one of the following:</p> <ul style="list-style-type: none"> <li>0 – error waiting for response to AT command</li> <li>1 – GSM modem responded with error to AT command</li> <li>2 - +CMS error</li> <li>3 - +CME error</li> <li>4 – transmission to GSM modem failed, serial port error</li> <li>5 - supplied PIN number is incorrect format</li> </ul>
<p>HS_SMSDLL_EV_SEND_FAIL</p>	<p>Send operation (HsSmsDIISendCs) failed.</p> <p>Arg1 – error code, one of the following:</p> <ul style="list-style-type: none"> <li>0 – error waiting for response to AT command</li> <li>1 – GSM modem responded with error to AT command</li> <li>2 - +CMS error</li> <li>3 - +CME error</li> <li>4 – transmission to GSM modem failed, serial port error</li> </ul> <p>Arg2 – indicates if COM port has been closed or not:</p> <ul style="list-style-type: none"> <li>0 – SMS port is still open, new Send, Read or Delete operations possible</li> <li>1 – SMS port is closed, port must be open again before any other operations are possible</li> </ul>
<p>HS_SMSDLL_EV_DELETE_FAIL</p>	<p>Delete operation (HsSmsDIIDeleteCs) failed.</p> <p>Arg1 – error code, one of the following:</p> <ul style="list-style-type: none"> <li>0 – error waiting for response to AT command</li> <li>1 – GSM modem responded with error to AT command</li> <li>2 - +CMS error</li> <li>3 - +CME error</li> <li>4 – transmission to GSM modem failed, serial port error</li> </ul> <p>Arg2 – indicates if COM port has been closed or not:</p> <ul style="list-style-type: none"> <li>0 – SMS port is still open, new Send, Read or Delete operations possible</li> <li>1 – SMS port is closed, port must be open again before any other</li> </ul>

	operations are possible
HS_SMSDLL_EV_READ_FAIL	<p>Read operation (HsSmsDllReadCs) failed.</p> <p>Arg1 – error code, one of the following:</p> <p>0 – error waiting for response to AT command</p> <p>1 – GSM modem responded with error to AT command</p> <p>2 - +CMS error</p> <p>3 - +CME error</p> <p>4 – transmission to GSM modem failed, serial port error</p> <p>6 – error parsing received SMS message</p> <p>7 – no message found at specified message storage index</p> <p>Arg2 – indicates if COM port has been closed or not:</p> <p>0 – SMS port is still open, new Send, Read or Delete operations possible</p> <p>1 – SMS port is closed, port must be open again before any other operations are possible</p>

*arg1*

- argument 1, the meaning of this parameter depends on the value of evt, see description of evt parameter

*arg2*

- argument 2, the meaning of this parameter depends on the value of evt, see description of evt parameter

*arg3*

- argument 3, the meaning of this parameter depends on the value of evt, see description of evt parameter

*lpStatus*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, this parameter is the address of ANSI byte buffer containing the string indicating the status of message in mobile device memory, "REC UNREAD" or "REC READ"

*status\_len*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, contains the length of status string

*lpSender*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, this parameter is the address of ANSI byte buffer containing GSM number of the sender of SMS message

*number\_len*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, this parameter is the length of sender number

*lpTimestamp*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, this parameter is the address of ANSI byte buffer which contains GSM 03.40 TP-Discharge-Time in time-string format: "yy/MM/dd, hh:mm:ss+zz", where characters indicate year (two last digits), month, day, hour, minutes, seconds and time zone. For example, 6th of May 1994, 22:10:00 GMT+2 hours equals "94/05/06,22:10:00+08"

*timestamp\_len*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, contains the length of timestamp string

*lpText*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, contains address of SMS message text buffer (ANSI byte array). The buffer may contain plain text ASCII characters or non printable binary data depending on text encoding specified in call to HsSmsDIIReadCs.

*msg\_text\_len*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, contains the length of SMS text buffer

*\*rc*

- Pointer to variable to receive function return code. See "Return values".

**Return values:**

HS_SMS_RC_NO_EVENTS	There are no events in the HsSmsDII event queue. All function parameters passed by reference must be ignored.
HS_SMS_RC_OK	An event has been read from HsSmsDII event queue, all function parameters passed by reference are valid

**Sample usage:**

'Read events from timer

```
Sub OnReadEvents()
  Dim evt As Long
  Dim arg1 As Long
  Dim arg2 As Long
  Dim arg3 As Long
  Dim status_len As Long
  Dim number_len As Long
  Dim msg_text_len As Long
  Dim timestamp_len As Long
  Dim rc As Long
  Dim errstrlen As Long
```

```
HsSmsDIIReadEventsVB Form1.ComboCom.ListIndex, ByVal VarPtr(evt), _
  ByVal VarPtr(arg1), ByVal VarPtr(arg2), ByVal VarPtr(arg3), _
  VarPtr(msg_status(1)), ByVal VarPtr(status_len), VarPtr(gsm_number(1)), _
  ByVal VarPtr(number_len), VarPtr(Timestamp(1)), ByVal VarPtr(timestamp_len), _
  VarPtr(msg_text(1)), ByVal VarPtr(msg_text_len), ByVal VarPtr(rc)
```

```

If rc = HS_SMS_RC_NO_EVENTS Then
Else
  Select Case evt
  Case HS_SMSDLL_EV_OPEN_OK
    Form1.Status.Caption = "SMS Port Open Ok. Configuration Complete"
    Form1.CommandOpen.Enabled = True
    Form1.CommandRead.Enabled = True
    Form1.CommandSend.Enabled = True
    Form1.CommandDelete.Enabled = True
    Form1.CommandOpen.Caption = "HsSmsDllClose"

  Case HS_SMSDLL_EV_SEND_OK
    Form1.CommandSend.Enabled = True
    Form1.CommandRead.Enabled = True
    Form1.CommandDelete.Enabled = True
    Form1.Status.Caption = "SMS Sent OK: msg reference " & arg1

  Case HS_SMSDLL_EV_DELETE_OK
    Form1.CommandSend.Enabled = True
    Form1.CommandRead.Enabled = True
    Form1.CommandDelete.Enabled = True
    Form1.Status.Caption = "SMS Deleted OK: msg index " & arg1

  Case HS_SMSDLL_EV_READ_OK
    Form1.CommandSend.Enabled = True
    Form1.CommandRead.Enabled = True
    Form1.CommandDelete.Enabled = True

    If arg3 = HS_SMS_RC_OK Then
      Form1.Status.Caption = "Message read complete: Success"
      msg_status_uc = StrConv(msg_status, vbUnicode)
      gsm_number_uc = StrConv(gsm_number, vbUnicode)
      Timestamp_uc = StrConv(Timestamp, vbUnicode)
      msg_text_uc = StrConv(msg_text, vbUnicode)

      Form1.MsgStatus.Caption = msg_status_uc
      Form1.Timestamp.Caption = Timestamp_uc
      Form1.TextGsmNumber.Text = gsm_number_uc
      Form1.Smstext.Text = msg_text_uc
    Else
      Form1.Status.Caption = "Message read failed: decryption error " & arg3
    End If

  Case HS_SMSDLL_EV_NEW_RECEIVED
    Form1.Status.Caption = "New SMS Received, index " & arg1 & " Click HsSmsDllRead to read"
    Form1.TextRIndex.Text = "" & arg1
    Form1.TextDIndex.Text = "" & arg1

  Case HS_SMSDLL_EV_OPEN_FAIL
    HsSmsDllGetErrVB arg1, VarPtr(errstr(1)), ByVal VarPtr(errstrlen)
    errstr_uc = StrConv(errstr, vbUnicode)

    Form1.Status.Caption = "SMS Port Open Failed " & errstr_uc
    Form1.CommandOpen.Enabled = True

  Case HS_SMSDLL_EV_SEND_FAIL

```

```

HsSmsDllGetErrVB arg1, VarPtr(errstr(1)), ByVal VarPtr(errstrlen)
errstr_uc = StrConv(errstr, vbUnicode)

Form1.Status.Caption = "SMS Send Fail: " & errstr_uc

If arg2 = SMS_OPEN Then
    Form1.CommandSend.Enabled = True
    Form1.CommandRead.Enabled = True
    Form1.CommandDelete.Enabled = True
Else
    Form1.CommandSend.Enabled = False
    Form1.CommandRead.Enabled = False
    Form1.CommandDelete.Enabled = False
    Form1.CommandOpen.Caption = "HsSmsDllOpen"
End If

Case HS_SMSDLL_EV_DELETE_FAIL
HsSmsDllGetErrVB arg1, VarPtr(errstr(1)), ByVal VarPtr(errstrlen)
errstr_uc = StrConv(errstr, vbUnicode)

Form1.Status.Caption = "SMS Delete Fail: " & errstr_uc

If arg2 = SMS_OPEN Then
    Form1.CommandSend.Enabled = True
    Form1.CommandRead.Enabled = True
    Form1.CommandDelete.Enabled = True
Else
    Form1.CommandSend.Enabled = False
    Form1.CommandRead.Enabled = False
    Form1.CommandDelete.Enabled = False
    Form1.CommandOpen.Caption = "HsSmsDllOpen"
End If

Case HS_SMSDLL_EV_READ_FAIL
HsSmsDllGetErrVB arg1, VarPtr(errstr(1)), ByVal VarPtr(errstrlen)
errstr_uc = StrConv(errstr, vbUnicode)

Form1.Status.Caption = "SMS Read Fail: " & errstr_uc

If arg2 = SMS_OPEN Then
    Form1.CommandSend.Enabled = True
    Form1.CommandRead.Enabled = True
    Form1.CommandDelete.Enabled = True
Else
    Form1.CommandSend.Enabled = False
    Form1.CommandRead.Enabled = False
    Form1.CommandDelete.Enabled = False
    Form1.CommandOpen.Caption = "HsSmsDllOpen"
End If

End Select
End If

```

### 2.3.7 HsSmsDllGetErrVB

#### Declaration:

Declare Function HsSmsDllGetErrVB Lib "HsSmsDll.dll" \_  
(ByVal err As Long, ByVal lpbuff As Long, ByRef bflen As Long) As Long

#### Summary:

Use this function to get a descriptive string, corresponding to HsSmsDll error code

#### Parameters:

*err*

- error code, returned in argument 1 (arg1) of HsSmsDllReadEventsVB function with events: HS\_SMSDLL\_EV\_OPEN\_FAIL, HS\_SMSDLL\_EV\_SEND\_FAIL, HS\_SMSDLL\_EV\_DELETE\_FAIL and HS\_SMSDLL\_EV\_READ\_FAIL. This error code provides additional information about the error. The error code is one of the following:

- 0 – error waiting for response to AT command
- 1 – GSM modem responded with error to AT command
- 2 - +CMS error
- 3 - +CME error
- 4 – transmission to GSM modem failed, serial port error
- 5 – Open failed, PIN supplied incorrect format
- 6 – error parsing received SMS message
- 7 – no message found at specified message storage index

*lpbuff*

- address of an ANSI byte buffer to receive the error string corresponding to integer error code, may be one of the following:

"No Response To AT Command"  
"Error Response to AT Command"  
"+CMS Error"  
"+CME Error"  
"Incorrect PIN format"  
"Error parsing read message"  
"Message not found"

*bflen*

- variable passed by reference which receives the number of characters in error string

#### Return values:

HS_SMS_RC_OK	Success, error string copied to lpbuff address
HS_SMS_RC_INV_PAR	Invalid parameters

#### Sample usage:

```
HsSmsDllGetErrVB arg1, VarPtr(errstr(1)), ByVal VarPtr(errstrlen)  
errstr_uc = StrConv(errstr, vbUnicode)
```

```
Form1.Status.Caption = "SMS Send Fail: " & errstr_uc
```

## 2.4 ActiveX COM Object Interface

### 2.4.1 SmsOpen

#### Declaration:

```
HRESULT SmsOpen(  
    [in] long portno,  
    [in] long baudrate,  
    [in] long databits,  
    [in] long parity,  
    [in] long stopbits,  
    [in] long use_fc,  
    [in, string] unsigned char *pin,  
    [in] long init_wait_secs,  
    [in] long enable_log,  
    [out] long *rc);
```

#### Summary:

Use this method to open and configure GSM device attached to specified serial port. This method must be called prior to calling other library functions.

#### Parameters:

##### *portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

##### *baudrate*

– Serial port rate in bauds, one of the following: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200, 128000, 256000

##### *databits*

– Number of data bits, one of the following: 5, 6, 7, 8

##### *parity*

– Parity code as follows: 2 - EVEN, 3 - MARK, 0 - NONE, 1 - ODD, 4 - SPACE

##### *stopbits*

– Stop bits code as follows: 0 – 1 Stop bit, 1 – 1.5 Stop bits, 2 – 2 Stop bits

##### *use\_fc*

– Set to 1 to use hardware RTS/CTS flow control

##### *pin*

– 4 digit SIM PIN in zero terminated ASCII string format. Set to NULL if your SIM does not require PIN.

##### *init\_wait\_secs*

– GSM hardware initialisation delay in seconds. This is the delay between asserting DTR on the serial port being opened and sending first AT configuration command to the GSM device. This delay may be necessary to allow a GSM module to initialise, before it accepts AT commands. Delay value depends on GSM module vendor. If delay is not required, set to 0.

##### *enable\_log*

- If set to 1, event log file called HsSms.log shall be created and all important events showing HsSms Library activity shall be logged in it. Set to 0 to disable event log

\*rc

– Pointer to variable to receive function return code. See “Return values”.

### Return values:

HS_SMS_RC_OK	Success, GSM device configuration is in progress, the actual result of operation determined by polling with method SmsReadEvents
HS_SMS_RC_BAD_PORT	Invalid port number
HS_SMS_RC_INV_PAR	Invalid parameter
HS_SMS_RC_PORT_ALR_OPEN	Port is already open
HS_SMS_RC_NO_MEM	No free memory
HS_SMS_RC_COM_OPEN_FAIL	Failed to open COM port
HS_SMS_RC_DLL_LOAD_ERROR	HsSmsDll.Dll could not be loaded
HS_SMS_RC_DLL_FUNC_ERROR	Function address within HsSmsDll could not be resolved

### Sample usage:

Please refer to source code in HsSmsActiveX\_demo folder provided with the evaluation version

## 2.4.2 SmsSend

### Declaration:

```
HRESULT SmsSend(  
    [in] long portno,  
    [in, string] unsigned char *smc_number,  
    [in, string] unsigned char *dest_number,  
    [in] long lpSmsText,  
    [in] long sms_text_len,  
    [in] long text_encoding,  
    [in] long is_encryption_on,  
    [in] long encryption_alg,  
    [in] long lpEncryptionKey,  
    [in] long encrypt_key_bits,  
    [out] long *rc);
```

### Summary:

Use this method to send SMS message. The port must be previously open with method SmsOpen

**Parameters:**

*portno*

- Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

*smcsc\_number*

- SMS Service Centre number via which the SMS message is to be sent, this number is different for each GSM network operator. The number is of String type

*dest\_number*

- Destination GSM number where the SMS message is to be sent. The number is of String type

*lpSmsText*

- Address of SMS message buffer to send. The HsSmsDll.dll receives the buffer as ANSI byte array. Depending on selected encoding (see parameter *text\_encoding*) the content of the buffer may be ASCII printable characters or non ASCII binary content.

*sms\_text\_len*

- length of SMS message buffer in bytes.

*text\_encoding*

- This parameter determines how the content of *sms\_text* buffer is encoded before it is sent to GSM module. This parameter can take of the following values:

SMSENC_ASCII	(0x0) The buffer <i>sms_text</i> is sent "as is" unmodified.
SMSENC_BINHEX	(0x1) Each byte from <i>sms_text</i> buffer is sent as 2 ASCII hex characters. For example the <i>sms_text</i> buffer contains the following binary data {0x01,0xFF,0xEB}. This buffer shall be encoded and sent as follows: "01FFEB" (string) or in hex representation 0x30,0x31,0x46,0x46,0x45,0x42  Using this method the maximum length of <i>sms_text</i> buffer must not exceed 80 bytes

*is\_encryption\_on*

- Set to 1 to encrypt the content of *sms\_text* buffer before sending. The text encoding shall be automatically forced to SMSENC\_BINHEX, regardless of the value of the *text\_encoding* parameter. The maximum length of source *sms\_text* buffer must not exceed 80 bytes. Set to 0 to send SMS text without encryption.

*encryption\_alg*

- If *is\_encryption\_on* is set to 1, this parameter specifies the encryption algorithm to be used, one of the following:

- 0 – AES
- 1 – DES
- 2 – 3DES
- 3 – ARC4,
- 4 – CAST128
- 5 – BLOWFISH
- 6 – TWOFISH

*lpEncryptionKey*

- If `is_encryption_on` is set to 1, this parameter is the address of buffer containing encryption key (must be ANSI byte array)

`encrypt_key_bits`

- If `is_encryption_on` is set to 1, this parameter specifies the length of encryption key in number of bits

`*rc`

– Pointer to variable to receive function return code. See “Return values”.

#### Return values:

HS_SMS_RC_OK	Success, SMS message is submitted for sending. The actual send result is determined by polling with method <b>SmsReadEvents</b>
HS_SMS_RC_NOT_INIT	HsSmsDll library not initialised
HS_SMS_RC_BAD_PORT	Invalid port number
HS_SMS_RC_PORT_NOT_OPEN	Specified port is not open
HS_SMS_RC_INV_STATE	Invalid internal state for sending SMS
HS_SMS_RC_INV_PAR	Invalid parameters
HS_SMS_RC_DLL_LOAD_ERROR	HsSmsDll.Dll could not be loaded
HS_SMS_RC_DLL_FUNC_ERROR	Function address within HsSmsDll could not be resolved

#### Sample usage:

Please refer to source code in HsSmsActiveX\_demo folder provided with the evaluation version

## 2.4.3 SmsReadEvents

#### Declaration:

```

HRESULT      SmsReadEvents(
    [in]      long    portno,           // COM port number, zero based
    [out]     long    *evt,           // event
    [out]     long    *arg1,          // argument 1
    [out]     long    *arg2,          // argument 2
    [out]     long    *arg3,          // argument 3
    [in]      long    lpStatus,        // new message status
    [out]     long    *status_len,     // new message status length
    [in]      long    lpSender_number, // GSM sender number
    [out]     long    *number_len,     // sender number length
    [in]      long    lpTimestamp,     // message timestamp
    [out]     long    *timestamp_len,  // timestamp length
    [in]      long    lpMsg_text,     // message text
    [out]     long    *msg_text_len,   // message text length

```

[out] long \*rc);

### Summary:

This method should be called periodically from a timer, to determine completion result of submitted commands and to detect the arrival of new SMS messages

### Parameters:

*portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

*evt*

- If return code is HS\_SMS\_RC\_OK, this variable contains HsSmsDll event. This variable is passed by reference and filled by HsSmsDll with an event read from internal event queue. If no events have occurred the return value of the function is HS\_SMS\_RC\_NO\_EVENTS.

The possible events are listed below:

<b>Event</b>	<b>Description</b>
HS_SMSDLL_EV_OPEN_OK	Open operation completed with success
HS_SMSDLL_EV_SEND_OK	SMS send operation completed with success, SMS message has been send by mobile device, Arg1 is GSM 03.40 TP-Message-Reference in integer format
HS_SMSDLL_EV_DELETE_OK	Delete operation completed with success – SMS message with specified message storage index has been deleted. Arg1 is message index of the deleted
HS_SMSDLL_EV_READ_OK	SMS read operation completed with success, and:  status – contains the string indicating the status of message in mobile device memory, "REC UNREAD" or "REC READ"  status_len – length of status string  sender_number – GSM number of the sender of SMS message  number_len – length of sender number  timestamp – contains GSM 03.40 TP-Discharge-Time in time-string format: "yy/MM/dd,hh:mm:ss+zz", where characters indicate year (two last digits), month, day, hour, minutes, seconds and time zone. For example, 6th of May 1994, 22:10:00 GMT+2 hours equals "94/05/06,22:10:00+08"  timestamp_len – length of timestamp string  msg_text – SMS message text buffer. The buffer may contain plain text ASCII characters or non printable binary data depending on text encoding specified in call to HsSmsDllReadCs.

	<p>msg_text_len – length of SMS text buffer</p> <p>Arg3 contains message read operation result code:  HS_SMS_RC_OK – message is read correctly</p> <p>HS_SMS_RC_ENCRYPT_ERR_ALG – decryption error, invalid encryption algorithm</p> <p>HS_SMS_RC_ENCRYPT_ERR_INVP – decryption error, invalid parameter</p> <p>HS_SMS_RC_ENCRYPT_ERR_KEYSZ - decryption error, invalid key size</p> <p>HS_SMS_RC_ENCRYPT_ERR – decryption error, no additional information</p>
HS_SMSDLL_EV_NEW_RECEIVED	<p>Mobile device has received new SMS message. Arg1 contains SMS message index in the mobile device storage. The message may be read from mobile device using method SmsRead</p>
HS_SMSDLL_EV_OPEN_FAIL	<p>Open operation failed.</p> <p>Arg1 – error code, one of the following:</p> <ul style="list-style-type: none"> <li>0 – error waiting for response to AT command</li> <li>1 – GSM modem responded with error to AT command</li> <li>2 - +CMS error</li> <li>3 - +CME error</li> <li>4 – transmission to GSM modem failed, serial port error</li> <li>5 - supplied PIN number is incorrect format</li> </ul>
HS_SMSDLL_EV_SEND_FAIL	<p>Send operation failed.</p> <p>Arg1 – error code, one of the following:</p> <ul style="list-style-type: none"> <li>0 – error waiting for response to AT command</li> <li>1 – GSM modem responded with error to AT command</li> <li>2 - +CMS error</li> <li>3 - +CME error</li> <li>4 – transmission to GSM modem failed, serial port error</li> </ul> <p>Arg2 – indicates if COM port has been closed or not:  0 – SMS port is still open, new Send, Read or Delete operations possible</p>

	<p>1 – SMS port is closed, port must be open again before any other operations are possible</p>
HS_SMSDLL_EV_DELETE_FAIL	<p>Delete operation failed.  Arg1 – error code, one of the following:  0 – error waiting for response to AT command</p> <p>1 – GSM modem responded with error to AT command  2 - +CMS error  3 - +CME error  4 – transmission to GSM modem failed, serial port error</p> <p>Arg2 – indicates if COM port has been closed or not:  0 – SMS port is still open, new Send, Read or Delete operations possible  1 – SMS port is closed, port must be open again before any other operations are possible</p>
HS_SMSDLL_EV_READ_FAIL	<p>Read operation failed.</p> <p>Arg1 – error code, one of the following:  0 – error waiting for response to AT command  1 – GSM modem responded with error to AT command  2 - +CMS error  3 - +CME error  4 – transmission to GSM modem failed, serial port error  6 – error parsing received SMS message  7 – no message found at specified message storage index</p> <p>Arg2 – indicates if COM port has been closed or not:  0 – SMS port is still open, new Send, Read or Delete operations possible  1 – SMS port is closed, port must be open again before any other operations are possible</p>

*arg1*

- argument 1, the meaning of this parameter depends on the value of evt, see description of evt parameter

*arg2*

- argument 2, the meaning of this parameter depends on the value of evt, see description of evt parameter

*arg3*

- argument 3, the meaning of this parameter depends on the value of evt, see description of evt parameter

*lpStatus*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, this parameter is the address of ANSI byte buffer containing the string indicating the status of message in mobile device memory, "REC UNREAD" or "REC READ"

*status\_len*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, contains the length of status string

*lpSender*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, this parameter is the address of ANSI byte buffer containing GSM number of the sender of SMS message

*number\_len*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, this parameter is the length of sender number

*lpTimestamp*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, this parameter is the address of ANSI byte buffer which contains GSM 03.40 TP-Discharge-Time in time-string format: "yy/MM/dd, hh:mm:ss+zz", where characters indicate year (two last digits), month, day, hour, minutes, seconds and time zone. For example, 6th of May 1994, 22:10:00 GMT+2 hours equals "94/05/06,22:10:00+08"

*timestamp\_len*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, contains the length of timestamp string

*lpText*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, contains address of SMS message text buffer (ANSI byte array). The buffer may contain plain text ASCII characters or non printable binary data depending on text encoding

*msg\_text\_len*

- If evt parameter is HS\_SMSDLL\_EV\_READ\_OK, contains the length of SMS text buffer

*\*rc*

- Pointer to variable to receive function return code. See "Return values".

#### Return values:

HS_SMS_RC_NO_EVENTS	There are no events in the HsSmsDll event queue. All function parameters passed by reference must be ignored.
HS_SMS_RC_OK	An event has been read from HsSmsDll event queue, all function parameters passed by reference are valid

#### Sample usage:

Please refer to source code in HsSmsActiveX\_demo folder provided with the evaluation version

## 2.4.4 SmsRead

### Declaration:

```
HRESULT SmsRead(  
    [in] long portno,           // COM port number, zero based  
    [in] long msg_index,       // index of message to read  
    [in] long text_encoding,   // message encoding (ascii or binary as Ascii Hex)  
    [in] long is_encryption_on, // 1= encryption enabled, 0=encryption disabled  
    [in] long encryption_alg,  // encryption algorithm  
    [in] long lpEncryptionKey, // encryption key  
    [in] long encrypt_key_bits, // encryption key size in bits  
    [out] long *rc);          // return code pointer
```

### Summary:

Use this method to read an already received SMS message from GSM device, given the message index of the message in mobile device storage.

### Parameters:

#### *portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

#### *msg\_index*

– Message index of SMS message to read in the mobile device storage.

#### *text\_encoding*

- This parameter determines how the content of sms buffer is decoded after it read from mobile device. This parameter can take of the following values:

SMSENC_ASCII	(0x0) The SMS buffer contains ASCII text and will be returned “as is” unmodified.
SMSENC_BINHEX	(0x1) Each byte in the SMS message buffer is represented with 2 ASCII hex characters. For example the buffer contains the following binary data: 0x30,0x31,0x46,0x46,0x45,0x42 or in other terms string “01FFEB” shall be decoded after reading as follows: {0x01,0xFF,0xEB}

#### *is\_encryption\_on*

- Set to 1 to decrypt the content of SMS buffer after reading. The text encoding shall be automatically forced to SMSENC\_BINHEX, regardless of the value of the *text\_encoding* parameter. Set to 0 to read SMS buffer without decryption.

#### *encryption\_alg*

- If *is\_encryption\_on* is set to 1, this parameter specifies the encryption algorithm to be used, one of the following:

- 0 – AES
- 1 – DES
- 2 – 3DES

- 3 – ARC4,
- 4 – CAST128
- 5 – BLOWFISH
- 6 – TWOFISH

*lpEncryptionKey*

- If *is\_encryption\_on* is set to 1, this parameter is the address of buffer containing encryption key (must be ANSI byte array)

*encrypt\_key\_bits*

- If *is\_encryption\_on* is set to 1, this parameter specifies the length of encryption key in number of bits

*\*rc*

– Pointer to variable to receive function return code. See “Return values”.

**Return values:**

HS_SMS_RC_OK	Success, the command to read SMS message is submitted. The actual result of read operation (an the SMS buffer read) should be obtained by polling with method <b>ReadEvents</b>
HS_SMS_RC_INV_PAR	Invalid parameters
HS_SMS_RC_NOT_INIT	HsSmsDll library not initialised
HS_SMS_RC_BAD_PORT	Invalid port number
HS_SMS_RC_PORT_NOT_OPEN	Specified port is not open
HS_SMS_RC_INV_STATE	Invalid internal state for reading SMS

**Sample usage:**

Please refer to source code in HsSmsActiveX\_demo folder provided with the evaluation version

## 2.4.5 SmsDelete

**Declaration:**

```
HRESULT SmsDelete(
    [in] long portno,           // COM port number, zero based
    [in] long msg_index,      // index of message to delete
    [out] long *rc);         // return code pointer
```

**Summary:**

Use this function to delete an SMS message from GSM device, given the message index of the message in mobile device storage.

**Parameters:**

*portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

*msg\_index*

– Message index of SMS message to delete in the mobile device storage.

*\*rc*

– Pointer to variable to receive function return code. See “Return values”.

**Return values:**

HS_SMS_RC_OK	Success, the command to delete SMS message is submitted. The actual result of delete operation should be obtained by polling with method <b>SmsReadEvents</b>
HS_SMS_RC_NOT_INIT	HsSmsDll library not initialised
HS_SMS_RC_BAD_PORT	Invalid port number
HS_SMS_RC_PORT_NOT_OPEN	Specified port is not open
HS_SMS_RC_INV_STATE	Invalid internal state for reading SMS

**Sample usage:**

Please refer to source code in HsSmsActiveX\_demo folder provided with the evaluation version

**2.4.6 SmsClose**

**Declaration:**

```

HRESULT SmsClose(
    [in] long portno,           // COM port number, zero based
    [out] long *rc);

```

**Summary:**

Use this method to close an open SMS port. This function releases all resources associated with the port and closes the specified COM port

**Parameters:**

*portno*

– Zero based Serial port COM device number to which GSM device is attached. 0 corresponds to COM1, 1 to COM2 and so on. Possible values are from 0 to 31 (supported COM devices from COM1 to COM32)

*\*rc*

– Pointer to variable to receive function return code. See “Return values”.

**Return values:**

HS_SMS_RC_OK	Success, SMS port closed
HS_SMS_RC_BAD_PORT	Invalid port number
HS_SMS_RC_INV_PAR	Invalid parameter
HS_SMS_RC_PORT_NOT_OPEN	Port is not open

**Sample usage:**

Please refer to source code in HsSmsActiveX\_demo folder provided with the evaluation version

## 2.4.7 SmsGetRcString

### Declaration:

```
HRESULT SmsGetRcString(  
    [in] long rc_code,  
    [in] long lpRcStr);
```

### Summary:

Use this method to get a descriptive string, corresponding to HsSmsDll return code

### Parameters:

rc\_code – return code

lpRcStr – long pointer to buffer to receive descriptive string

### Sample usage:

Please refer to source code in HsSmsActiveX\_demo folder provided with the evaluation version

## 2.5 Error Codes and Other Constants

### 2.5.1 HsSmsDll return codes

Name	Value	Comment
HS_SMS_RC_OK	1	Success
HS_SMS_RC_BAD_PORT	2	invalid port number
HS_SMS_RC_PORT_ALR_OPEN	3	port already open
HS_SMS_RC_NO_MEM	4	No free memory
HS_SMS_RC_COM_OPEN_FAIL	5	Failed to open serial port
HS_SMS_RC_COM_TX_FAIL	6	Failed to send data over serial port
HS_SMS_RC_START_TIMER_FAIL	7	failed to start timer
HS_SMS_RC_INV_PAR	8	invalid parameters
HS_SMS_RC_NOT_INIT	9	library not initialised
HS_SMS_RC_PORT_NOT_OPEN	10	SMS port not open
HS_SMS_RC_INV_STATE	11	SMS engine state is not valid for this operation
HS_SMS_RC_NO_EVENTS	12	no new events in event queue
HS_SMS_RC_ENCRYPT_ERR	13	encryption error, reason unspecified
HS_SMS_RC_ENCRYPT_ERR_INV_P	14	encryption error, invalid parameters
HS_SMS_RC_ENCRYPT_ERR_ALG	15	encryption error, invalid algorithm
HS_SMS_RC_ENCRYPT_ERR_KEYSZ	16	encryption error, invalid key size
HS_SMS_RC_DLL_LOAD_ERROR	17	Failed to load HsSmsDll.dll
HS_SMS_RC_DLL_FUNC_ERROR	18	Failed to resolve a function address in HsSmsDll.dll

### 2.5.2 HsSmsDll event notifications

Name	Value	Comment
HS_SMSDLL_EV_OPEN_OK	0	SMS port open completed OK
HS_SMSDLL_EV_SEND_OK	1	SMS sending completed OK (message sent)
HS_SMSDLL_EV_DELETE_OK	2	SMS deleting completed OK (message deleted)
HS_SMSDLL_EV_READ_OK	3	SMS read completed OK (message read)

HS_SMSDLL_EV_NEW_RECEIVED	4	New SMS received notification
HS_SMSDLL_EV_OPEN_FAIL	5	SMS port open failed
HS_SMSDLL_EV_SEND_FAIL	6	SMS sending failed
HS_SMSDLL_EV_DELETE_FAIL	7	SMS deleting failed
HS_SMSDLL_EV_READ_FAIL	8	SMS reading failed
HS_SMSDLL_EV_READ_SMSC_FAIL	9	Read operation of SMSC number (service centre number) failed
HS_SMSDLL_EV_SMSC_READ_OK	10	SMSC number (service centre number) has been successfully retrieved from the mobile device and can be read in using functions HsSmsDIIReadSmsc (C Sharp) or HsSmsDIIReadSmscVB (Visual Basic)

### 2.5.3 HsSmsDII Extra Error Codes

Name	Value	Comment
HS_SMSDLL_EV_EX_AT_TIMEOUT	0	timeout waiting for response to AT command
HS_SMSDLL_EV_EX_AT_ERROR	1	modem responded with error to configuration command
HS_SMSDLL_EV_EX_CMS_ERROR	2	+CMS ERROR
HS_SMSDLL_EV_EX_CME_ERROR	3	+CME ERROR
HS_SMSDLL_EV_EX_COMTX	4	transmission to modem failed
HS_SMSDLL_EV_EX_PIN_FORMAT	5	Open failed, PIN supplied incorrect format
HS_SMSDLL_EV_EX_MSG_FORMAT	6	Error parsing received message
HS_SMSDLL_EV_EX_MSG_NOTFOUND	7	No message found at specified index

### 2.5.4 Other Constants

Name	Value	Comment
SMS_OPEN	0	port is open
SMS_CLOSED	1	port is closed
SMSENC_ASCII	0	Plain ASCII characters
SMSENC_BINHEX	1	Binary data, each byte is transmitted as two ASCII hex bytes