vsmsgw C#
Calling vsmsgw.dll from C#
Revision: 2.00
Date: 08-16-2013

vsmsgw C#

Calling vsmsgw.dll from C#

Ver. 2.00

Table of contents

Introduction

This package shows how vsmsgw.dll can be called from a C# application.

The VSMSGW library is a small library that encapsulates some of the transactions that is possible to make against a RTCU unit. For a description of the complete Gateway protocol, please consult the appropriate document from Logic IO.

Contents of package:

Windows Application	The Visual Studio 2010 project for the example program.
RTCU Application	A .VPL application which can be used to test the application
vsmsgw CSharp.pdf	This document.

The whole package can be tested on a pc by installing the following applications

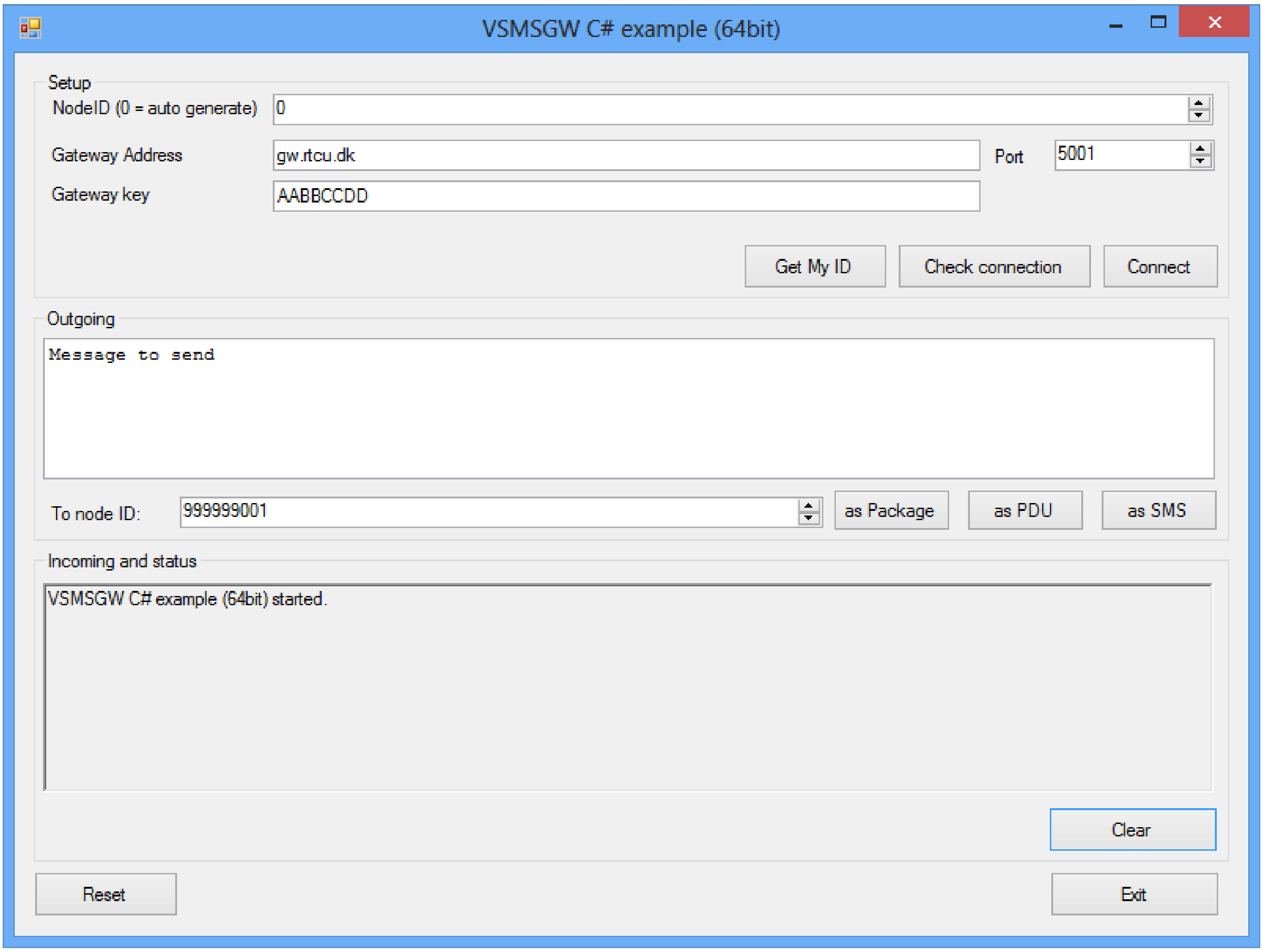
- RTCU Gateway 2
    - Install service on local machine with defaults
    - Or use our public server at gw.rtcu.dk
- RTCU IDE
    - To simulate the RTCU unit
    - In the simulator check the GPRS setup to ensure it connect to the gateway.

Notes about 32/64 bits systems

The application will try to copy the correct version of the “vsmsgw.dll” and its dependencies to the application directory when started, if this does not succeed a warning dialog will be shown and existing files will be used if present.

C# example project

The example project is a simple demonstration of how to call vsmsgw.dll to send and receive VSMS messages to/from RTCU units.



NodeID (0 = auto generate): 0

Gateway Address: gw.rtcu.dk

Port: 5001

Gateway key: AABBCCDD

Message to send

To node ID: 999999001

VSMSGW C# example (64bit) started.

The application is found within the ‘Windows Application’ project folder.

The Required dll’s are found in the release folder of the project, and the latest version can also be downloaded from http://www.logicio.com as a part of the “RTCU Gateway Deployment Package”.

How to send messages:

First select the server to connect to, and the press the “connect” button. The application will now try to connect to the gateway.

To check if a connection has been obtained press the “Check connection” button, and the status will be written to the “Incoming and status” area.

To send a SMS

To send a text message (SMS), type the receivers node id into the “To node ID” field, the entered default is the default of the RTCU IDE simulator. Then write the text and press the “as SMS” button.

If the include VLP application is running in the target unit, the text will be echoed back, and appear in the “Incoming and status” area in the form:
SMS received from ***<Node ID>*** size ***<package length>***: ***<text>***

To send a PDU

To send a binary message (PDU), type the receivers node id into the “To node ID” field, the entered default is the default of the RTCU IDE simulator. Then write the text and press the “as PDU” button.

The text will now be converted to a byte array and send as binary data to the target unit.

If the include VLP application is running in the target unit, the data will be echoed back, and appear in the “Incoming and status” area in the form:
PDU received from ***<Node ID>*** size ***<package length>***

To send a data package

To send a binary data package over the gateway, type the receiver’s node id into the “To node ID” field, the entered default is the default of the RTCU IDE simulator. Then write a text and press the “as PDU” button.

The text will now be converted to a byte array and send as binary data to the target unit.

If the include VLP application is running in the target unit, the data will be echoed back, and appear in the “Incoming and status” area in the form:
Data packet received from ***<Node ID>*** size ***<package length>***

Receiving SMS/Data

All data received will be displayed in the “Incoming and status” area, depending on the callback function which has received the data, will appear in one of these forms

Text over GSM

SMS received from ***<Node ID>*** size ***<package length>***: ***<text>***

Data over GSM

PDU received from ***<Node ID>*** size ***<package length>***

Data over gateway

Data packet received from ***<Node ID>*** size ***<package length>***

Functions exposed from “VSMSGW.CS”

The static class VSMSGW implemented in “vsmsgw.cs” explosed the raw functions from the vsmsgw.dll library by defining the import statements acording to the vsmsgw.h file.

For a detail description on the inplementation of the vsmsgw packet please consult the “RTCU Gateway Deployment Package”

The Library is a collection of the following functions described in this section.

Return Codes

Symbolic name	Value	Description
GWRC_OK	0	Operation successful
GWRC_ERROR	1	Unspecified error
GWRC_NOT_CON	2	Not connected
GWRC_TIMEOUT	3	A timeout occurred
GWRC_INV_LEN	5	Invalid length
GWRC_IS_OPEN	7	Is already open
GWRC_NOT_OPEN	8	Is not open
GWRC_INV_PARM	9	Invalid parameters
GWRC_DST_UNREACH	10	Destination node is unreachable

Init

Declaration

```
static public extern GWRC Init(
        UInt32 MyNodeID,
        String GWIP,
        Int32 GWPort,
        String GWKey,
        cbfuncText SMSText,
        cbfuncPDU SMSPDU,
        IntPtr arg);
```

Description

Initialize the connection to the RTCU Gateway, and setup callback function for SMS and PDU.

Callbacks must be defined according to the defined delegates

Input

MyNodeId	The nodeid for the PC application. If set to 0, it will be assigned by the GPRS Gateway
GWIP	The IP address (or symbolic name) of the GPRS Gateway
GWPort	The portnumber the GPRS Gateway listens on
GWKey	The key value (an 8 character password) used to access the GPRS Gateway.
SMSText	A callback function that will be called whenever an Text SMS is received
SMSPdu	A callback function that will be called whenever an PDU SMS is received
Arg	A user supplied 32 bit variable.

Reply

GWRC_IS_OPEN, GWRC_OK

InitAdv

Declaration

```
static public extern GWRC InitAdv(
        UInt32 MyNodeID,
        String GWIP,
        Int32 GWPort,
        String GWKey,
        [MarshalAs(UnmanagedType.LPArray, SizeConst = 16)] Byte[] CryptKey);
```

Description

Initialize the connection to the GPRS Gateway.

Input

MyNodeId	The nodeid for the PC application. If set to 0, it will be assigned by the GPRS Gateway (0 for automatic)
GWIP	The IP address (or symbolic name) of the GPRS Gateway
GWPort	The port number the GPRS Gateway listens on
GWKey	The key value (an 8 character password) used to access the GPRS Gateway.
CryptKey	The encryption key used to access the GPRS Gateway. (16 Bytes)

Reply

GWRC_IS_OPEN, GWRC_OK

SetDataPackageCB

Declaration

static public extern void SetDataPackageCB(cbfuncPackage PACKAGE, IntPtr arg);

Description

Set the callback function that will be called when a data package is received.

Callback must be defined according to the defined data delegate

Input

PACKAGE	A callback function that will be called whenever a data package is received
arg	A user supplied 32 bit variable

SetSMSPDUCB

Declaration

static public extern void SetSMSPDUCB(cbfuncPDU SMSPDU, IntPtr arg);

Description

Set the callback function that will be called when a PDU SMS is received.

Callback must be defined according to the defined data delegate

Input

SMSPDU	A callback function that will be called whenever a PDU SMS is received
arg	A user supplied 32 bit variable

SetSMSTextCB

Declaration

static public extern void SetSMSTextCB(cbfuncText SMSText, IntPtr arg);

Description

Set the callback function that will be called when a PDU SMS is received.

Callback must be defined according to the defined text delegate

Input

SMSText	A callback function that will be called whenever a Text SMS is received
arg	A user supplied 32 bit variable

IsConnected

Declaration

static public extern Boolean IsConnected();

Description

Determine connection status to the GPRS Gateway.

Input

None.

Reply

False if not connected
True if connected

GetMyNodeID

Declaration

static public extern GWRC GetMyNodeID(ref UInt32 MyNodeID);

Description

This function will return this nodes node id. This function is used especially when a dynamic node id is requested (by setting NodeID to 0 before connecting).

Input

MyNodeId	The node id for the PC application

Reply

GWRC_OK, GWRC_NOT_CON, GWRC_NOT_OPEN

SendSMS

Declaration

```
static public extern GWRC SendSMS(
        UInt32 HisNodeID,
        [MarshalAs(UnmanagedType.LPStr)] String str,
        ref Int32 rc);
```

Description

Send a Text SMS message to the specified NodeID, the return code from the RTCU unit, will be contained in rc

Input

HisNodeID	The node number (serial number) of the receiving RTCU unit
str	The string to send to the receiving RTCU unit. Maximum size is 160 characters.
rc	The return code from the receiving RTCU unit, 0 if OK, else error

Reply

GWRC_OK, GWRC_ERROR, GWRC_DST_UNREACH, GWRC_NOT_CON, GWRC_NOT_OPEN, GWRC_INV_LEN, GWRC_INV_PARM

SendPDU

Declaration

```
static public extern GWRC SendPDU(
        UInt32 HisNodeID,
        [MarshalAs(UnmanagedType.LPArray)] Byte[] data,
        Int32 length,
        ref Int32 rc);
```

Description

Send a PDU SMS message to the specified NodeID, the return code from the RTCU unit, will be contained in rc.

Input

HisNodeID	The node number (serial number) of the receiving RTCU unit
data	The data to send to the receiving RTCU unit Maximum size is 140 bytes.
length	The length of data to send
rc	The return code from the receiving RTCU unit, 0 if OK, else error

Reply

GWRC_OK, GWRC_ERROR, GWRC_DST_UNREACH, GWRC_NOT_CON, GWRC_NOT_OPEN, GWRC_INV_LEN, GWRC_INV_PARM

SendDataPackage

Declaration

```
static public extern GWRC SendDataPackage(
        UInt32 HisNodeID,
        [MarshalAs(UnmanagedType.LPArray)] Byte[] data,
        Int32 length,
        ref Int32 rc);
```

Description

Send a data package to the specified NodeID, the return code from the RTCU unit, will be contained in rc.

Input

HisNodeID	The node number (serial number) of the receiving RTCU unit
data	The data to send to the receiving RTCU unit Maximum size is 480 bytes.
length	The length of data to send
rc	The return code from the receiving RTCU unit, 0 if OK, else error

Reply

GWRC_OK, GWRC_ERROR, GWRC_DST_UNREACH, GWRC_NOT_CON, GWRC_NOT_OPEN, GWRC_INV_LEN, GWRC_INV_PARM

Callbacks

The actual declarations in “VSMSGW.CS” take care of proper call to/from the unmanaged code, and allowing auto creation byte arrays.

Declarations
Callbacks must bed implemented according to these declarations.

---

cbfuncText

Declaration

```
public delegate UInt16 cbfuncText(
        Int32 HisNodeID,
        [MarshalAs(UnmanagedType.LPStr)] String str,
        IntPtr arg
        );
```

Description

Callbacks function for receiving text messages.

An example on implementation can be found in “MainForm.cs”.

Input

HisNodeID	The node id of the sender (the serial number of the RTCU that sent the message)
str	The text string sent by the RTCU. Max size is 160 characters.
arg	A user supplied 32 bit variable that was set when callback was set.

Reply

(Will be send to sender), 0 if you accept the data package else 1.

cbfuncPDU

Declaration

```
public delegate UInt16 cbfuncPDU(
        int HisNodeID,
        [MarshalAs(UnmanagedType.LPArray, SizeParamIndex = 2)] byte[] data,
        int length,
        IntPtr arg
        );
```

Description

Callback function for receiving DPU / data SMS.

An example on implementation can be found in "MainForm.cs".

Input

HisNodeID	The node id of the sender (the serial number of the RTCU that sent the message)
data	The data sent by the RTCU. Max size is 140 bytes.
length	The length of data sent by the RTCU
arg	A user supplied 32 bit variable that was set when callback was set.

Reply

(Will be send to sender), 0 if you accept the data package else 1.

cbfuncPackage

Declaration

```
public delegate UInt16 cbfuncPackage(
        Int32 HisNodeID,
        [MarshalAs(UnmanagedType.LPArray, SizeParamIndex = 2)] Byte[] data,
        Int32 length,
        IntPtr arg
        );
```

Description

Callback function for receiving data packages.

An example on implementation can be found in “MainForm.cs”.

Input

HisNodeID	The node id of the sender (the serial number of the RTCU that sent the message)
data	The data sent by the RTCU. Max size is 480 bytes.
length	The length of data sent by the RTCU
arg	A user supplied 32 bit variable that was set when callback was set.

Reply

(Will be send to sender), 0 if you accept the data package else 1.