FF-AUTOMATION

User Manual

v.1.02

DB2SMS

Updated 6th Jun 2008

Program description:

Program sends SMS messages (in database) to defined phone number (in database).
It also receives SMS messages through GSM modem (in serial port).

1 DB2SMS version history

v.1.0

- 5.1.2007 (ArMe)
- Initialization of GSM modem
- Sending SMS messages (to one, to group, fault events messages)
- Receiving SMS messages (report, history and event message types)
- Program status in database
- Compatible with different databases

2 DB2SMS

2.1 Needed components etc.

- Indusoft Web Studio SCADA (Or some other SCADA with ODBC support*)
- DB2SMS program
- Win2000/XP OS
- Microsoft .Net Framework 2.0 (can be downloaded from Microsoft homepage)
- GSM-modem
- Database, which support ODBC –interface (MySQL, PostgreSQL)
- Note! Indusoft web studio demo uses PostgreSQL Database connection.

* Please contact FF-Automation for more information!

2.2 Installing DB2SMS

1. Check that you have installed MS .Net Framework 2.0 (This can be checked form control panel Add/remove programs list) http://www.microsoft.com/downloads/details.aspx?familyid=0856eacb-4362-4b0d-8edd-aab15c5e04f5&displaylang=en
2. Copy DB2SMS program (DB2SMS.exe) and configuration file (config.ini) to your PC. (e.g. C:\DB2SMS directory).
3. Installation is ready

2.3 Configuring Program

2.3.1 General configuration description

When configuring the program you should check the following parameters:

- PIN code
- The serial port which is used by GSM modem
- The location of the database, DNS name, user name and password

1. Open *config.ini* with text editor. The file looks like this:

```
Modem_Port='2'
Modem_PIN='1234'
Modem_Speed='9600'
Modem_Handshaking='3'
; ConnectionString='DRIVER={MySQL ODBC 3.51
Driver};SERVER=localhost;DATABASE=smsindusoft;USER=username;PASSWORD
=passw;'
ConnectionString='DRIVER={PostgreSQL
ANSI};SERVER=localhost;DATABASE=smsindusoft;USER=username;PASSWORD=
passw;'
```

2. Used serial port for modem is defined in the first line.
3. SIM –card's PIN number is defined in the second line. You can leave this undefined if the *PIN code request* is configured as 'off' in your SIM card.
4. Modem speed is defined in the third line. Normally this is 9600, so normally you do not need to change it.
5. Modem_Handshaking parameter is important and it should be defined correctly. If you are unsure which value to use, you can try one by one which work correctly. Look the table next table for more information!
6. The connection string defines the database connection. MySQL connection string is commented. Change the user name and password to correct ones.

- DB2SMS uses ODBC –driver to create connection with database.
- MySQL
  - Uses MyODBC database connection driver. This can be downloaded for example:
  - http://dev.mysql.com/downloads/connector/odbc/3.51.html
  - ConnectionString should include Server, Database, User and Password –parameters (look example file)
- PostgreSQL
  - PostgreSQL includes needed ODBC connection drivers. You don't need to install these separately.
  - ConnectionString should include Server, Database, User and Password –parameters (look example file)
- Access
  - Access 2000 –database is not supported. If you need to use this, you should use Access2SMS program, not DB2SMS program. Note that DB2SMS program has more features than Access2SMS.
- Other databases
  - Other databases are not tested with DB2SMS. In principle all the databases that uses ODBC interface should work without problems.

7. Normally you don't need to give any other parameters to DB2SMS, but in the next table includes all the parameters that can be configured.

Parameter	Mandatoryn/Optional	Description
ConnectionString='x'	Mandatory	Database connection definition
Modem_Port='x'	Mandatory	Used serial port for GSM modem
Modem_Speed	Mandatory	Serial port speed (normally 9600)
Modem_Handshaking	Mandatory	Handshaking level 0 = No handshaking 1 = XOn / XOff 2 = Request to send 3 = Request to send and XOn / XOff
pin='xxxx'	Optional	SIM –card's PIN –code
license='move'	Optional	Activates licence move state
loglevel='x'	Optional (default 1)	Selects the level for message log: 1:Received and send messages are written to log (default). 2:The serial communication is written to log. 3: Both 1 and 2.

Config.ini Parameter explanations

Note! If your PIN code request is enabled in your SIM card, you should not change the PIN code while the program is running. DB2SMS tries to initialize SIM card with the defined PIN code, and if the code is wrong, the SIM card probably gets locked.

2.3.2 Troubleshooting - Modem initialization problems

If you have problems with initializing modem you can check that these are ok:

- You have inserted SIM card
- PIN code request is disabled / you have right PIN code in config.ini
- You have right type of cable and it is in right COM port
- Baudrate is set correctly (9600)

If the above were ok you can test that the modem is in TEXT mode and the ECHO is OFF.

Open HyperTerminal and start session with parameters COM#, 9600,8/1/N,OFF

Command	Modem should Reply
AT	OK
AT+CPIN?	OK or +CPIN: SIM PIN or +CPIN: READY If you get ERROR or +CPIN: SIM PUK, there's a Problem with the SIM.
AT+CMGF?	1 (=text mode)
ATE0	OK (sets echo off)

3 DB2SMS

3.1 Starting the program

Program can be started by double-clicking the *DB2SMS.exe* file. If your program is unlicensed, the program informs how much evaluation time you have left and when clicking the *Continue Evaluate software* –button the application is started. If the configuration of the program is correct you should see small “mobile phone” picture in the right down corner (system tray):

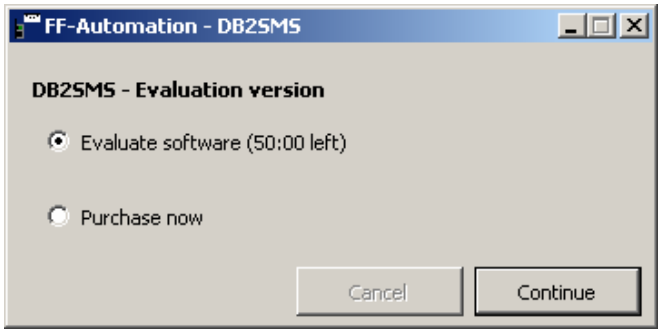
3.2 Closing the program

Program can be closed by double clicking the mouse’s **right** button on the “mobile phone” picture in the right down corner (system tray). After that you will get confirmation message and answering ‘Yes’ closes the program.

3.3 Program license

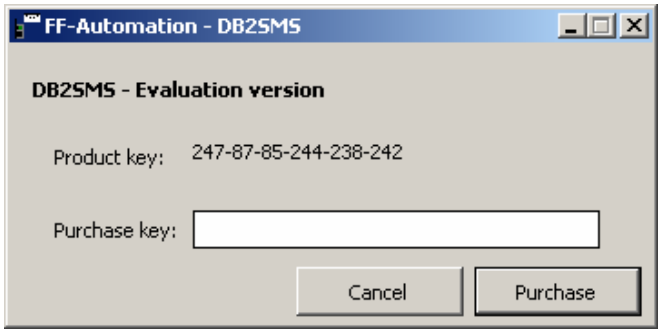
3.3.1 Evaluation period

DB2SMS program can be used 50 hours evaluation time without licensing. Program inform how much time is left when it is started.



3.3.2 Purchase and registration

Program can be licensed by selecting the *Purchase now* checkbox and clicking *Continue*. Program generates product key, which is unique key for PC. This key should then be send to FF-Automation (e.g. by e-mail) and FF-Automation then sends back the right *Purchase key* to activate the license. Program informs if the activation were successful.



3.3.3 Moving the license

The license can be moved to other PC. The original license is removed.

You should first install DB2SMS to destination PC. After that you should add the following line to the original already licensed PC’s configuration file (Config.ini)

license=’move’

Look the previous Config .ini parameter explanations –table!

This activates the license transfer state. Write the Product key –field the destination PC’s Product key and click Move. Program warns about the license transfer, if you answer ‘Yes’ the original license is removed and you get new Purchase key to be inserted to destination PC’s Purchase key –field.

Note! Be careful when you insert the product key. If the product key is faulty, the program doesn't generate right purchase key and you will lose the license. If this happens please contact with FF-Automation. (Include the product key and the generated purchase key in your message.)

4 Receiving message

Program can receive three types of messages: report-, history data- and event messages.

The program goes through the *Phones_Receiving* table in the database. Only the messages which are coming from the active phone numbers (enabled <>0) are accepted.

4.1 Event message (*Received_Events*)

Event message can be used to send e.g. alarm, warning, status change or any other freely defined message from the GSM-PLC. Message consists of Message type identifier (A), Event type (nn as integer), DateTime (DDMMYYHHMMSS) and Actual message.

4.1.1 Format

Ann,DDMMYYHHMMSS,Message

4.1.2 Examples

Receiving event message "Problem" with date and time (ID 0):

A0,060705150000,Problem

Receiving event message "Problem" without date and time (ID 1):

A1,,Problem

Received events are stored in Received_Events –table. Look more from database structure chapter!

4.2 History (*Received_History*)

History message can be used to send one variables history e.g. some sensor's measurement value history. History message uses Variable_Crosslinks –table. Message consists of Message type identifier (H), variable number (nn as integer), DateTime (DDMMHHYYMMSS), Sample Interval (HHMM) and actual data (D1,D2…) The data part starts from the most recent value (D1), the oldest value comes last.

4.2.1 Format

Hnn,DDMMYYHHMMSS,HHMM,D1,D2,D3,D4…

4.2.2 Examples

Receiving history message from variable number 1, Sample interval 1 hour with DateTime:

H1,060705150000,0100,20,19,19,18,18,19,17,15,14,13,12,12

Receiving history message from variable number 2, Sample interval 3 hour 30 min without DateTime:

H2,0230,20,19,19,18,18,19,17,15,14,13,12,12

Received history messages are stored in Received_History –table. The data is divided in separate rows in the table. The variable number is changed to variable name using Variable_Crosslinks -table. Look more from database structure chapter!

4.3 Report (*Received_Report*)

Report message can be used to send the latest values of many variables. Report message consists of Message type identified (D), first variable number (nn as integer), DateTime (DDMMHHYYMMSS) and actual data (Variable values).

Program uses *Variable_Crosslink* –table to change the variable number to variable name. Every variable in the messages data part increases variable's number by one so that first variable's V1 number is nn, V2 is nn+1…

4.3.1 Format

D*nn,DDMMYYHHMMSS,V1,V2,V3,V3…*

4.3.2 Examples

Receiving Report message from variable values numbers 0,1 and 2 with DateTime:

D0,060705150000,20,19,21

Receiving Report message from variable values numbers 1,2 and 3 without DateTime:

D1,,20,19,21

5 Sending message

Message sending can be done in two ways: To one phone number (Send to one) or to Group of phone numbers (Send to group).

5.1 Send to Group messages

Program sends all messages in the *SendAll* table which has flag = 1 value to phone numbers defined in *Phones_Sending* table which are active (enabled =1).

5.2 Send to One messages

Program sends all messages in the *Send_One* table which has flag =1. Before sending the message user should write value to *Phone* –field and after that change the flag to 1 which sends the message.

5.3 Error messages

Server program needs to send e.g. increasing counter value to *PRG_Counter* –field. If the program detects that this value has not changed and the time in *A2SMS_Counte*r has increased to *Counter_Max* it sends defined message to phone number and put flag to zero. When the value in the *PRG_Counte*r –field changes it changed flag to 1.

6 Error codes

6.1 Log files

There are three log files which can be used to debug the possible fault situations etc. The logging level is configured in Config.ini file.

File name	description

sms_traffic.txt serial_traffic.txt state_log.txt	SMS traffic Serial traffic Program states (Error codes)

6.2 Error Codes

In addition to state_log, DB2SMS stores its states also to database (*Errormessage* - table).

Error code	Description
0	No errors
Modem Errors (write to log and database)	
M01x M02x M03x M04x M05x M06y, M07x M08x M1x M2x M3x	Could not open serial port Message couldn't be send to serial port Modem is not answering PIN code error (wrong PIN / SIM locked) Could not go to Text mode GSM modem signal strenght information sending disabling failed Unknown problem when initialization modem Error in receiving message Error in sending message Error in calculating new messages
Critical Errors (write to log and database (if possible), info to screen and stops the program)	
C00 C02 C03	Critical program error (Unknown error – report to FF-Automation!) PIN code insertion error Database not found
Errors in message handling (write to log only)	
P0 P1 P2 P3	DateTime format incorrect Report message format incorrect Message format incorrect Error in saving message

7 Database stucture

- **ErrorMessage (Programs writes)**
  - Errorcode (String)
    - Look Error codes chapter!
- **Phonebook_Sending (User defines)**
  - Name (String)
    - Program doesn't use this field. It can be used in SCADA program.
  - Phone (String)
    - Phone number
  - Enabled (Integer)
    - 0: Phone number not in use
    - 1: Phone number in use
- **Phonebook_Receiving (User defines)**
  - Name (String)
    - Program reads the field to Received tables. Field can be used also in SCADA program.
  - Phone (String)
    - Phone number (Needs land code e.g. +358401234)
  - Enabled
    - 0: Phone number not in use
    - 1: Phone number in use

- **Received_Events Program writes)**
  - Mindex (Integer, increasing)
    - Unique identifier for every received message
  - Flag (Integer)
    - Value 1 indicates that the message is new. The user program (e.g. SCADA) can reset this value to zero to indicate that it has read the message. Old messages can be deleted e.g. after reading /once per day or week to protect that the table size doesn't increase too much.
  - Date (String)
    - Message receiving date (yyyy-mm-dd)
  - Time (String)
    - Message receiving time (hh:mm:ss)
  - Phone (String)
    - Phone number from where the message was received
  - Name (String)
    - Sender's name (copied from *Phonebook_Receiving* -table)
  - Event (String)
    - Actual message, "free format"
  - ID (Integer)
    - Event ID
- **Received_Report (Program writes)**
  - Mindex (Integer, increasing)
    - Unique identifier for every received message
  - Flag (Integer)
    - Value 1 indicates that the message is new. The user program (e.g. SCADA) can reset this value to zero to indicate that it has read the message. Old messages can be deleted e.g. after reading /once per day or week to protect that the table size doesn't increase too much.
  - Date (String)
    - Message receiving date (yyyy-mm-dd)
  - Time (String)
    - Message receiving time (hh:mm:ss)
  - Phone (String)
    - Phone number from where the message was received
  - Name (String)
    - Sender's name (copied from *Phonebook_Receiving* -table)
  - Variable (String)
    - Variable's name. DB2SMS copies this name from *Report_Crosslinks* –table's name field using the variable number.
  - Value (String)
    - Variable's value
    - String type. Supports both text and number formats. This allows variable's value to be send also in text format.
- **Received_History**
  - Mindex (Integer, increasing)
    - Unique identifier for every received message
  - Flag (Integer)
    - Value 1 indicates that the message is new. The user program (e.g. SCADA) can reset this value to zero to indicate that it has read the message. Old messages can be deleted e.g. after reading /once per day or week to protect that the table size doesn't increase too much.

        - Date (String)
            - Message receiving date (yyyy-mm-dd)
        - Time (String)
            - Message receiving time (hh:mm:ss)
        - Phone (String)
            - Phone number from where the message was received
        - Name (String)
            - Sender's name (copied from *Phonebook_Receiving* -table)
        - Variable (String)
            - Variable's name (Read from the message)
        - Value (String)
            - Variable's value
- **Send_All (User defines, program changes)**
        - Mindex (Integer, increasing)
            - Unique identifier for every send message
        - Flag (Integer)
            - Send message –flag. This is reset to zero after sending the message.
        - Message (String)
            - Send message
            - 160 characters
- **Send_One (User defines, program changes)**
        - Mindex (Integer, increasing)
            - Unique identifier for every send message
        - Flag (Integer)
            - Send message –flag. This is reset to zero after sending the message.
        - Phone (String)
            - Phone number
        - Message (String)
            - Send message
            - 160 characters
- **Variable_Crosslinks (User defines)**
        - Index (Integer)
            - Variables index number
        - Name (String)
            - Variable's name
            - Note that e.g. Web Studio SCADA doesn't support special characters or spaces in this field!
- **Send_Watchdog (User defines partly, program changes)**
        - Name (String)
            - Identifier name field for e.g. SCADA application use. Not used by DB2SMS.
        - Enabled (Integer)
            - Watchdog enabled when 1, user defines.
        - Flag (Integer)
            - Changes to 1, when PRG_Counter and Old_PRG_Counter has different values for too long time.
            - (1=proram ok, 0=problem in program message send)
        - PRG_Counter (Integer)
            - SCADA should change this using e.g. increasing counter.
        - Old_PRG_Counter (Integer)
            - ~10 sec intervals the value from PRG_Counter is copied to this
        - A2SMS_Counter (Integer)

- Mikäli PRG_Counter ja Old_PRG_Counter olivat samoja, tämän arvo nousee (kasvaa yhdellä noin 10 sekunnin välein). Tämän arvo nollataan jos PRG_Counter ja Old_PRG_Counter ovat erisuuria.
- If PRG_Counter and Old_PRG_Counter has same values, the value will be increased by one (~10 seconds intervals). If they differ this value is reset to zero.

- Counter_Max (Integer)
  - Max value for A2SMS_Counter before sending the message and flag reset to zero.
- Phone (String)
  - Phone number to where the warning message is send
- Message (String)
  - Send warning message ”free format”

8 Support

Please contact FF-Automation by e-mail or call if you need additional support.

www.ff-automation.com