SMS Gateway

# Overview Schema

The interactions of SMS Gateway Server and related components can be seen on following picture.

incoming request from client

Decrypt & check signature

N

Is Valid?

N

is there any job to be proceed?

Delay every X second

X = Based on configuration

Y

Y

Request Collection

Worker Pool Manager

Y

is there any serial port in idle state?

Y

Execute the job

GSM Communication Library

Action Provider

It's return OK?

**GSM device**

Y

Plugin Manager

GSM Logging Database

Another Plugin

Database

# SMS Gateway Server Operation

The SMS Gateway Server operation consist of several steps

1. Read confirguration
2. Initialization port.
3. Establish connection.
4. Main loops is executed until it is signaled to be terminated.
5. Listen incoming data from TCP/IP.
6. Check for new request command in database.
7. Check for data integrity security (configure by PublicKey)
8. Check for received messages (configure by intervalReadMessage)
9. Execute job (configure by intervalWorkerQueue)
10. Closing connection and releasing resources.

# Installation Guide

SMS Gateway Server enables you to use your computer to send and receive SMS messages to and from mobile devices. It allow SMS messaging in two ways: directly connecting to the server application or inserting a command to the database using GSMClient library (to facilitate the sending/generate data to the server gateway).

For SMS Messaging with a GSM modem, you need to use suitable GSM device. While developing this application we used Wavecom hardware, however there should be no problem when using another GSM modem series. Make sure that you have install the GSM driver. (please check the website given by the device if available).

Hardware requirements:

1. CPU: Intel P4 3Ghz
2. RAM: 2 Gb
3. Harddisk 500 MB
4. Port: USB or RS232
5. Network if you use IP GSM device

Software requirements

1. Windows XP
2. Windows 7
3. Windows 8.0/8.1

The following should be installed:

1. .NET Framework 4.0
2. Install the GSM device driver

## Installation Steps

### Database

Just copy-paste the following script into SQL Server Management Studio

USE [SMSGW]

GO

/\*\*\*\*\*\* Object: Table [dbo].[WeeklyTrigger] Script Date: 04/17/2015 20:04:38 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[WeeklyTrigger](

[ID] [varchar](10) NOT NULL,

[RecursEvery] [int] NOT NULL,

[Sunday] [int] NOT NULL,

[Monday] [int] NOT NULL,

[Tuesday] [int] NOT NULL,

[Wednesday] [int] NOT NULL,

[Thursday] [int] NOT NULL,

[Friday] [int] NOT NULL,

[Saturday] [int] NOT NULL,

CONSTRAINT [PK\_WeeklyTrigger] PRIMARY KEY CLUSTERED

(

[ID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[ResultWorkItem] Script Date: 04/17/2015 20:04:38 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[ResultWorkItem](

[SeqNbr] [varchar](20) NOT NULL,

[Response] [text] NULL,

[CreatedDate] [datetime] NOT NULL

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[Quiz] Script Date: 04/17/2015 20:04:38 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[Quiz](

[Group] [int] NOT NULL,

[Keyword] [varchar](1000) NOT NULL,

[Response] [varchar](1000) NOT NULL

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[QueueWorkItem] Script Date: 04/17/2015 20:04:38 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[QueueWorkItem](

[SeqNbr] [varchar](20) NOT NULL,

[Command] [varchar](max) NOT NULL,

[ScheduleID] [varchar](20) NOT NULL,

[Created] [datetime] NOT NULL,

[Enabled] [bit] NOT NULL,

[RecursPoint] [datetime] NULL,

[LastExecuted] [datetime] NULL,

[NextExecuted] [datetime] NULL,

[Status] [varchar](1) NOT NULL,

CONSTRAINT [PK\_QueueWorkItem] PRIMARY KEY CLUSTERED

(

[SeqNbr] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[PrefixPhoneName] Script Date: 04/17/2015 20:04:38 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[PrefixPhoneName](

[ID] [varchar](3) NOT NULL,

[PrefixPhoneName] [varchar](50) NOT NULL,

CONSTRAINT [PK\_PrefixPhoneName] PRIMARY KEY CLUSTERED

(

[ID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[PhoneNumber] Script Date: 04/17/2015 20:04:38 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[PhoneNumber](

[IDContact] [varchar](20) NOT NULL,

[IDPrefixPhoneName] [varchar](3) NOT NULL,

[PhoneNumber] [varchar](50) NOT NULL,

CONSTRAINT [PK\_PhoneNumber] PRIMARY KEY CLUSTERED

(

[IDContact] ASC,

[IDPrefixPhoneName] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[Outbox] Script Date: 04/17/2015 20:04:38 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[Outbox](

[Sender] [varchar](30) NOT NULL,

[SeqNbr] [varchar](20) NOT NULL,

[Receiver] [varchar](870) NOT NULL,

[Message] [varchar](max) NOT NULL,

[Time] [datetime] NOT NULL,

[SenderOperator] [varchar](30) NOT NULL,

[DeviceResponse] [varchar](500) NULL,

[NetworkStatus] [varchar](100) NULL,

[Error] [varchar](200) NULL,

CONSTRAINT [PK\_Outbox\_1] PRIMARY KEY CLUSTERED

(

[SeqNbr] ASC,

[Receiver] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[MonthlyTrigger] Script Date: 04/17/2015 20:04:38 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[MonthlyTrigger](

[ID] [varchar](10) NOT NULL,

[Days] [varchar](50) NOT NULL,

[January] [int] NOT NULL,

[February] [int] NOT NULL,

[March] [int] NOT NULL,

[April] [int] NOT NULL,

[May] [int] NOT NULL,

[June] [int] NOT NULL,

[July] [int] NOT NULL,

[August] [int] NOT NULL,

[September] [int] NOT NULL,

[October] [int] NOT NULL,

[November] [int] NOT NULL,

[December] [int] NOT NULL,

CONSTRAINT [PK\_MonthlyTrigger] PRIMARY KEY CLUSTERED

(

[ID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[MessageStatus] Script Date: 04/17/2015 20:04:38 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[MessageStatus](

[Phonenumber] [varchar](870) NOT NULL,

[SeqNbr] [varchar](20) NOT NULL,

[Status] [varchar](1) NOT NULL,

[Source] [varchar](10) NOT NULL,

CONSTRAINT [PK\_MessageStatus\_1] PRIMARY KEY CLUSTERED

(

[Phonenumber] ASC,

[SeqNbr] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[Inbox] Script Date: 04/17/2015 20:04:38 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[Inbox](

[Sender] [varchar](30) NOT NULL,

[SeqNbr] [varchar](20) NOT NULL,

[Message] [varchar](max) NOT NULL,

[Time] [datetime] NOT NULL,

[ReceiverOperator] [varchar](30) NOT NULL,

[IsRead] [int] NULL,

CONSTRAINT [PK\_Inbox\_1] PRIMARY KEY CLUSTERED

(

[Sender] ASC,

[SeqNbr] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[DailyTrigger] Script Date: 04/17/2015 20:04:38 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[DailyTrigger](

[ID] [varchar](10) NOT NULL,

[RecursEvery] [int] NOT NULL,

CONSTRAINT [PK\_DailyTrigger] PRIMARY KEY CLUSTERED

(

[ID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[ContactName] Script Date: 04/17/2015 20:04:38 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[ContactName](

[ID] [varchar](20) NOT NULL,

[Name] [varchar](50) NOT NULL,

[Note] [varchar](2000) NULL,

CONSTRAINT [PK\_Contact] PRIMARY KEY CLUSTERED

(

[ID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

/\*\*\*\*\*\* Object: Table [dbo].[Answer] Script Date: 04/17/2015 20:04:37 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

SET ANSI\_PADDING ON

GO

CREATE TABLE [dbo].[Answer](

[KeyID] [varchar](100) NOT NULL,

[Description] [varchar](3000) NOT NULL,

CONSTRAINT [PK\_Answer] PRIMARY KEY CLUSTERED

(

[KeyID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

### Server Application

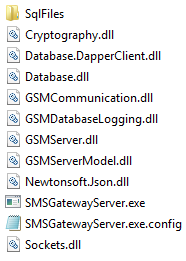
There is no setup wizard available right now, all you have to do is just copy bin folder into your

destination computer

The SMS Server files consist:

* SqlFiles – folder contains sql files.
* Cryptography.dll – used to encrypt/decrypt each communication between client and server.
* Database – it’s abstraction layer to hook-up micro ORM.
* Database.DapperClient.dll – it’s Dapper ORM implementation.
* GSMCommunication.dll – contains PDU Converters and basic functionallity of GSM device (SMS, Call, Phonebook, General features)
* GSMDatabaseLogging.dll – it’s a plugin that responsible to record received/send SMS messages.
* GSMServer.dll – it’s library that periodically scans for received/send messages, perform execution command received from database or TCP/IP.
* GSMServerModel.dll – it’s model data for sending command to the server.
* SMSGatewayServer.exe – it’s console application used to initiate GSMServer.
* SMSGatewayServer.exe.config – it’s XML configuration used to configure SMS Gateway Server.
* Socket.dll – it’s TCP monitor used to listen incoming data.

Here related components can be seen following picture



### Configuration

Before you run the server, please check the configuration SMSGatewayServer.exe.config

Please see the section SerialPorts, you have to change the serial port name depends on your GSM device. (See the Device Manager 🡪 Ports (COM & LPT))

<?xml version="1.0"?>

<configuration>

<configSections>

<section name="general" type="GSMServer.Configuration.GeneralSettings, GSMServer"/>

<section name="serialPorts" type="GSMServer.Configuration.SerialPortSettings, GSMServer"/>

<section name="plugin" type="GSMServer.Configuration.PluginSettings, GSMServer"/>

</configSections>

<!--

defaultPort - The port used client to connect the server.

defaultIP - The IP Address used client to connect the server

if the PC is connected to the network it will have 2 IP Address, 127.0.0.1 (localhost) and the IP address of network.

defaultEncoding - The Encoding used to convert string .

countryCode - The Country code of your Network Telecommunication .

smsSignature - The signature used to indicates the sender request if valid.

prefixOwnNumber - The prefixOwnNumber used to get the phonenumber of SIM card.

intervalWorkerQueue - The Interval used by server to check queue of request.

intervalReadMessage - The Interval used by server to read incoming message.

-->

<general defaultPort="13005" defaultIP="" defaultEncoding="ibm850" countryCode="62" smsGWSignature="TMMIN" prefixOwnNumber="own number" intervalWorkerQueue="3000" intervalReadMessage="3000"/>

<!-- example

<SerialPort

portName="COMxxxx" The port to use (for example, COM1).

baudRate="xxxxxx" The baud rate. (Type: System.Int32)

Parity="" One of the Parity values. see http://msdn.microsoft.com/en-us/library/system.io.ports.parity%28v=vs.110%29.aspx

dataBits="" The data bits value. (Type: System.Int32)

stopBits="" One of the StopBits values. see http://msdn.microsoft.com/en-us/library/system.io.ports.stopbits%28v=vs.110%29.aspx

handshake="" Handshaking protocol for serial port transmission of data. see http://msdn.microsoft.com/en-us/library/system.io.ports.handshake%28v=vs.110%29.aspx

ServiceCenter="62xxxxxxx" When you use Handphone/Cellphone as a modem, please specify the Service Center of the SIM Card.

pduMode="true/false" SMS Mode, it will PDU Mode or Text Mode

/>-->

<serialPorts>

<serialPort>

<add portName="COM3" baudRate="115200" parity="None" stopBits="One" dataBits="8" handshake="requestToSend" pduMode="true"/>

</serialPort>

</serialPorts>

<!—-

Plugin is used to listening every received and send data from the server to the client

assemblyFile - the name of assembly.

type - the name of class to be load.

-->

<plugin>

<assembly>

<add assemblyFile="GSMDatabaseLogging" type="GSMDatabaseLogging.Logging"/>

</assembly>

</plugin>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/></startup>

<connectionStrings>

<add name="SMSGW" providerName="System.Data.SqlClient" connectionString="Server=.;Database=SMSGW;User ID=you username;Password=your password;Trusted\_Connection=false;"/>

</connectionStrings>

<appSettings>

<!—-

SQLFolder - location of sql files

PublicKey - It’s encryption key used to decrypt every incoming data (32 byte)

-->

<add key="SQLFolder" value="SqlFiles"/>

<add key="PublicKey" value="0, 36, 0, 0, 4, 128, 0, 0, 148, 0, 0, 0, 6, 2, 0, 0, 0, 36, 0, 0, 82, 83, 65, 49, 0, 4, 0, 0, 1, 0, 1, 0, 235, 154, 120, 189, 24, 181, 220, 20, 179, 177, 213, 21, 175, 214, 184, 55, 144, 45, 41, 140, 227, 123, 3, 178, 132, 197, 217, 25, 44, 154, 213, 246, 210, 21, 225, 3, 219, 119, 88, 132, 244, 175, 168, 224, 199, 184, 98, 229, 66, 231, 83, 61, 89, 95, 225, 218, 145, 197, 202, 149, 100, 7, 32, 235, 46, 129, 120, 110, 131, 113, 109, 170, 197, 16, 255, 174, 240, 118, 130, 99, 101, 126, 107, 9, 75, 189, 64, 56, 178, 86, 40, 11, 56, 90, 160, 247, 209, 57, 84, 151, 47, 224, 42, 50, 126, 251, 182, 55, 3, 172, 224, 227, 163, 54, 184, 215, 13, 130, 29, 198, 73, 80, 157, 69, 230, 54, 97, 168"/>

<add key="ClientSettingsProvider.ServiceUri" value=""/>

</appSettings>

<system.web>

<membership defaultProvider="ClientAuthenticationMembershipProvider">

<providers>

<add name="ClientAuthenticationMembershipProvider" type="System.Web.ClientServices.Providers.ClientFormsAuthenticationMembershipProvider, System.Web.Extensions, Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" serviceUri=""/>

</providers>

</membership>

<roleManager defaultProvider="ClientRoleProvider" enabled="true">

<providers>

<add name="ClientRoleProvider" type="System.Web.ClientServices.Providers.ClientRoleProvider, System.Web.Extensions, Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" serviceUri="" cacheTimeout="86400"/>

</providers>

</roleManager>

</system.web>

</configuration>

Another part you have to be concern is general section, you can see following code:

<general

defaultPort="13005"

defaultIP=""

defaultEncoding="ibm850"

countryCode="62"

**smsGWSignature**="TMMIN"

prefixOwnNumber="own number"

**intervalWorkerQueue**="3000"

**intervalReadMessage**="3000"/>

if you have a lot of GSM modem and you want to use them all, you have to add the following configuration in SerialPorts Section, you just mention the name of the COM in your computer

<serialPorts>

<serialPort>

<add portName="COM3" baudRate="115200" parity="None" stopBits="One" dataBits="8" handshake="requestToSend" pduMode="true"/>

<add portName="YOUR-COM-NAME" baudRate="115200" parity="None" stopBits="One" dataBits="8" handshake="requestToSend" pduMode="true"/>

...

...

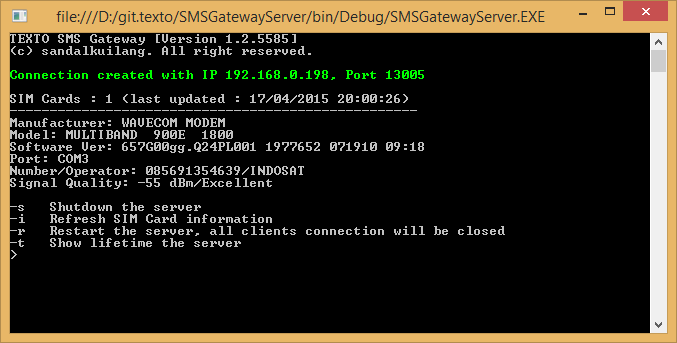
<add portName="YOUR-COM-NAME " baudRate="115200" parity="None" stopBits="One" dataBits="8" handshake="requestToSend" pduMode="true"/>

</serialPort>

</serialPorts>

Once you have successfully configure SMS Gateway

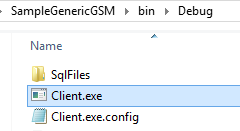
You can **run** the application SMSGatewayServer.exe and the console application will show up.



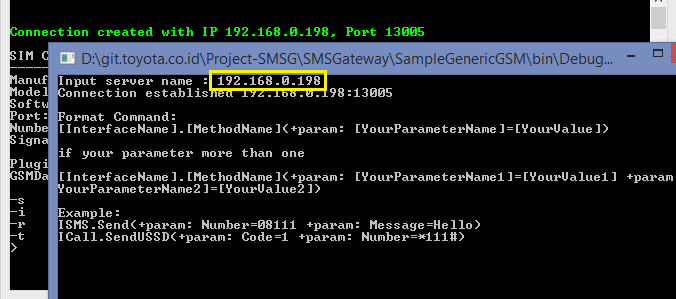
There are several command options.

1. –s used to closing all connection port and releasing resources.
2. –I used to refresh SIM Card information.
3. –r used to restart application.
4. –t used to show how long this application has been running.

to ensure sms gateway server is running properly ( sending/reading sms , call, send USSD code, etc. ), you can use the application **SampleGenericGSM/bin/Debug/Client.exe** , to test the TCP connection , and then sends a command to the server sms gateway.



Enter IP Address in accordance with the IP address that created by the server, as the following picture:



There are 36 command available on the server:

1. **ISMS.Send(+param: number={0} +param: message={1})**

a. parameter:

* + message: string contain text to be send
  + number: the number of recipient

b. example:

* + ISMS.Send(+param: message=lorem ipsum +param: number=0888881111)
  + ISMS.Send(+param: number=0888881111 +param: message=lorem ipsum )

1. **ISMS.Read()**
2. **ISMS.ReadAll()**
3. **ISMS.Delete(+param: index={0})**

a. parameter:

* + index: index of message on SIM Card

1. **ISMS.DeleteAll()**
2. **ICall.Dial(+param: number={0})**

a. parameter:

* + number: the number to dial

b. example:

* + ISMS.Dial(+param: number=0888881111)

1. **ICall.Hang(+param: number={0})**

a. parameter:

* + number: the number to hangup

b. example:

* + ISMS.Hang(+param: number=0888881111)

1. **ICall.Answer()**
2. **ICall.SendUSSD(+param: code=1 +param: number={0})**

a. parameter:

* + number: the number to USSD
  + code: 1 is return value, always set code=1

b. example:

* + ICall.SendUSSD(+param: code=1 +param: number=\*888#)

1. **IPhoneBook.FindPhoneBook(+param: memory={0} +param: name={1})**

a. parameter:

* + memory: the phonebook memory storage

• "**SM**": ADN Abbreviated Dialing Numbers (SIM phonebook)

• "**FD**": FDN Fixed Dialing Numbers (SIM restricted phonebook)

• "**ON**": MSISDN (SIM own numbers)

• "**EN**": ECC Emergency Call Codes (SIM or ME)

• "**LD**": LND Last Number Dialed

• "**MC**": ME missed (unanswered received) calls list

• "**ME**": ME phonebook

• "**MT**": combined ME and SIM phonebook (ME+SM)

• "**RC**": ME received calls list

• "**SN**": SDN Service Dialing Numbers (SIM special service numbers)

* + name: Searched pattern string (depends on the format of data stored in the phonebooks)

b. example:

* + IPhoneBook.FindPhoneBook(+param: memory=SM +param: name=yudha)

1. **IPhoneBook.ReadPhoneBook(+param: memory={0} +param: fromIndex={1} +param: toIndex={2})**

a. parameter:

* + memory: the phonebook memory storage
  + fromIndex: start index of contacts
  + toIndex: end index of contactsb.

b. example:

* + IPhoneBook.ReadPhoneBook(+param: memory=SM +param: fromIndex=1 +param: toIndex=10)

1. **IPhoneBook.GetInfo()**
2. **IPhoneBook.SetPhoneBookMemory(+param: memory={0})**

a. parameter:

* + memory: the phonebook memory storage

b. example:

* + IPhoneBook.SetPhoneBookMemory(+param: memory=SM)

1. **IPhoneBook.WritePhoneBook(+param: memory={0} +param: index={1} +param: name={2} +param; phoneNumber={3})**

a. parameter:

* + memory: the phonebook memory storage
  + index: the index of phonebook storage.
  + name: the name/alias that contains alphabet of phone number.
  + phoneNumber: the number

b. example:

* + IPhoneBook. WritePhoneBook(+param: memory=ON +param: index=1 +param: name=yudha +param: phoneNumber=088880)

1. **IGeneral.SetFunctionality(+param: level={0})**

This function used to set mode your GSM device (disable or enable)

a. parameter:

* + level: 0 set to minimum functionality, 1 to full functionality.

b. example:

* + IGeneral.SetFunctionality(+param: level =1)

1. **IGeneral.GetRegistrationStatus()**
2. **IGeneral.GetSignalQuality()**
3. **IGeneral.SetErrorMessageFormat(+param: format={0})**

This command disables or enables the use of the “+CME ERROR: <xxx>” or

“+CMS ERROR:<xxx>” result code instead of simply “ERROR”.

a. parameter:

* + format: 0 to disable ME error reports, use only « ERROR »,

1 to enable +CME ERROR: XXX or +CMS ERROR: XXX

b. example:

* + IGeneral.SetErrorMessageFormat(+param: format=1)

1. **IGeneral.GetPossibleCharacterSet()**
2. **IGeneral.GetCharacterSet()**
3. **IGeneral.SetCharacterSet(+param: characterSet={0})**

This command informs the ME which character set is used by the TE. The ME

can convert each character of entered or displayed strings.

a. parameter:

* + characterSet: one of the following value

“**GSM**” GSM default alphabet.

“**PCCP437**” PC character set code page 437.

“**CUSTOM**” User defined character set (cf. +WCCS

command).

“**HEX**” Hexadecimal mode. No character set used ; the

b. example:

* + IGeneral.SetCharacterSet(+param: characterSet=PCCP437)

1. **IGeneral.GetErrorMessageFormat()**
2. **IGeneral.GetManufacturer()**
3. **IGeneral.GetServiceCenter()**
4. **IGeneral.GetSoftwareVersion()**
5. **IGeneral.GetModelInformation()**
6. **IGeneral.GetIMSI()**
7. **IGeneral.GetOperator()**
8. **IGeneral.GetActivityStatus()**
9. **IGeneral.GetSerialNumber()**
10. **IGeneral.GetHardwareVersion()**
11. **IConfiguration.GetRawXML(+param: section={0})**

Return server application configuration in XML format

a. parameter:

* + section: one of the following value

general – contains general configuration of the sms gateway server

serialPorts - contains serialPorts section of the sms gateway server

plugin – contains plugin section of the sms gateway server

b. example:

* + IConfiguration. GetRawXML(+param: section=general)
  + IConfiguration. GetRawXML(+param: section=serialports)
  + IConfiguration. GetRawXML(+param: section=plugin)

1. **IConfiguration.SetRawXML(+param: section={0} +param: raw={1})**

a. parameter:

* + section: one of the following value
  + raw: contain value in XML format

b. example:

* + IConfiguration.SetRawXML(+param: section=general +param: raw=<general defaultPort="13005" defaultIP="" defaultEncoding="ibm850" countryCode="62" smsGWSignature="TMMIN" prefixOwnNumber="own number" intervalWorkerQueue="3000" intervalReadMessage="3000"/>)

1. **IConfiguration.Save()**
2. **IServer.GetRegisteredPlugins()**
3. **IServer.GetAvailableConnections()**

# Developer Guide

SMS Gateway can be used to send SMS messages from ASP.NET/Winform Application. There are two options you can choose to add functionality to your project to use an SQL SMS gateway architecture. In this architecture the application uses database server (Mircrosoft SQL Server, Oracle or MySql) for data storage. In this case the SQL SMS gateway configuration will work the following way:

Database

GSMClient

**SMS Gateway**

Mobile User

SMS



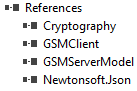
Figure 1 – SQL SMS Gateway configuration

There are three database tables. One will be used for received messages (Inbox), the second will be used to collecting commands (QueueWorkItem) and another will be used when sms has ben sent (Outbox).

If an SMS messages received by the system SMS Gateway will insert a record into Inbox, you application can read the Inbox table and check for incoming messages.

If you want to execute command (Send messages) or anything else you can used GSMClient to generate encrypted data and then insert to QueueWorkItem table. (SMS Gateway cannot recognize the data that not encrypted by GSMClient)

Here the example to use this SQL SMS Gateway:

1. Remember, Add this library to your project:
2. Write the following code:

string formattedSMS = string.Format(GSMClient.Command.CommandCollection.SMSSend, phoneNumber, message);

string result = GSMClient.Client.CreateCommand("Your-Signature", formattedSMS);

SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["SMSGW"].ConnectionString);

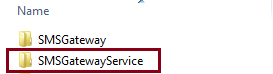
SqlCommand command = new SqlCommand(string.Format(“Your-Insert-Query”, result), connection);

connection.Open();

command.ExecuteNonQuery();

connection.Close();

You can see a complete example of SQL SMS Gateway in **ASP.NET Web Service** project - in the folder **SMSGatewayService**



the other option you can choose to communicate with the SMS gateway directly using TCP/IP. This option is deal If you don’t have database server attached to the server or yo dont want use it. The advantages to this solution is that it is faster reaction times. If you are receiving an SMS message using an TCP/IP SMS Gateway configuration, you can react faster with a response SMS.

GSMClient

**SMS Gateway**

Mobile User

SMS

TCP/IP

request

response



Figure 2 – TCP/IP SMS Gateway configuration for local network solution

It was designed to help developer be more productive and to be able to add SMS Functionality to any project efficiently. To send an SMS message you can submit it using TCP/IP request, here the following code to send SMS or another command:

1. Create class of ApplicationSettingKeySym

internal class ApplicationSettingKeySym : IKeySym

{

public byte[] GetKey()

{

string[] keys = “YOUR-PUBLIC-KEY /// this is your public key, you can read it from configuration.

byte[] publicKey = new byte[keys.Length - 1];

for (int i = 0; i < keys.Length - 1; i++)

{

publicKey[i] = Convert.ToByte(keys[i]);

}

return publicKey;

}

}

1. Create an Interface IGatewayService

public interface IGatewayService : INotifyPropertyChanged

{

bool Connected { get; }

string IPAddress { get; set; }

int Port { get; set; }

bool Connect();

bool Disconnect();

Task<string> ExecuteAsync(string command);

}

1. Create class GatewayService, and copy the following code

public class GatewayService : IGatewayService

{

private IClient client;

private Crypter crypter;

private bool isConnected;

private string ipAddress;

private int port;

public event PropertyChangedEventHandler PropertyChanged;

public GatewayService()

{

isConnected = false;

crypter = new Cryptography.Crypter(new ApplicationSettingKeySym());

}

public bool Connected

{

get

{

return isConnected;

}

}

public string IPAddress

{

get

{

return this.ipAddress;

}

set

{

this.ipAddress = value;

NotifyPropertyChanged("IPAddress");

}

}

public int Port

{

get

{

return this.port;

}

set

{

this.port = value;

NotifyPropertyChanged("Port");

}

}

/// <summary>

/// Start connecting to gateway sever

/// </summary>

/// <param name="ipAddress">the specified local IP address</param>

/// <param name="port">the port number</param>

public bool Connect()

{

try

{

if (string.IsNullOrEmpty(ipAddress))

{

throw new ArgumentNullException("IPAddress");

}

if (port == 0)

{

throw new ArgumentNullException("Port");

}

client = new GSMClient.Client(this.ipAddress, this.port, new ApplicationSettingKeySym());

client.Open();

isConnected = true;

return true;

}

catch (System.Net.Sockets.SocketException)

{

isConnected = false;

return false;

}

}

/// <summary>

/// Disconnect from gateway server

/// </summary>

public bool Disconnect()

{

client.Close();

isConnected = false;

return true;

}

/// <summary>

/// Send a command to gateway and wait the result.

/// </summary>

/// <param name="command">see GSMClient.Command.CommandCollection</param>

/// <returns>the result is in JSON format</returns>

public Task<string> ExecuteAsync(string command)

{

if ((isConnected) && (!string.IsNullOrEmpty(command)))

{

client.Send(command);

StringBuilder decrypt = new StringBuilder();

byte[] read;

string removeUnusedCharacter;

string decryptedResult;

TcpClient tcpClient = client.GetTcpClient();

NetworkStream networkStream = tcpClient.GetStream();

while (true)

{

if (!tcpClient.Connected)

break;

read = new byte[tcpClient.ReceiveBufferSize];

networkStream.Read(read, 0, read.Length);

removeUnusedCharacter = Encoding.GetEncoding("ibm850").GetString(read).Replace("\0", "");

decryptedResult = crypter.Decrypt(removeUnusedCharacter);

decrypt.Append(decryptedResult);

break;

}

return new Task<string>(() => (decrypt.ToString()));

}

return new Task<string>(() => ( string.Empty));

}

private void NotifyPropertyChanged([CallerMemberName] string propertyName = "")

{

if (PropertyChanged != null)

{

PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

1. Here the code to use GatewayService

IGatewayService service = new GatewayService();

service.Connect();

service.ExecuteAsync(string.Format(GSMClient.Command.CommandCollection.SMSSend, "08111000", "hello world"));

service.Disconnect();

### USE PLUGIN TO PROCESSING INCOMING MESSAGES

If you want to process incoming SMS messages as soon as they arrive to your system, you may want to create plugin in SMS Gateway.

1. Just create library with the following format:

public class Logging

{

public void Main()

{

// Your code here

}

public void OnDataReceived(string data)

{

// Your code here

}

public void OnDataSent(string data)

{

// Your code here

}

public void OnDisposed()

{

// Your code here

}

}

* void Main

it’s starting method, doesn’t return anything. Called when plugin loaded for the first time.

* void OnDataReceived(string)

this entry called when SMS Gateway Server received command.

* void OnDataSent(string)

this entry called when SMS Gateway Server executed command.

* void OnDisposed()

called when main application is shutdown.

1. Register your plugin

See your configuration file, and find Plugin section, change the assemblyFile and type

<plugin>

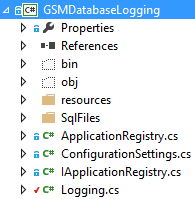
<assembly>

<add assemblyFile="Assembly-Name-without-extension" type="Namespace.Class-Name"/>

</assembly>

</plugin>

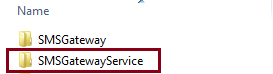
Here the example of Plugin, GSMDatabaseLogging



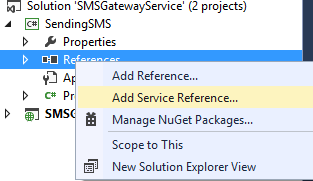
1. Place your Plugin library along with SMS Gateway Server application folder.

### Sending SMS Message Through ASP.NET Web Service

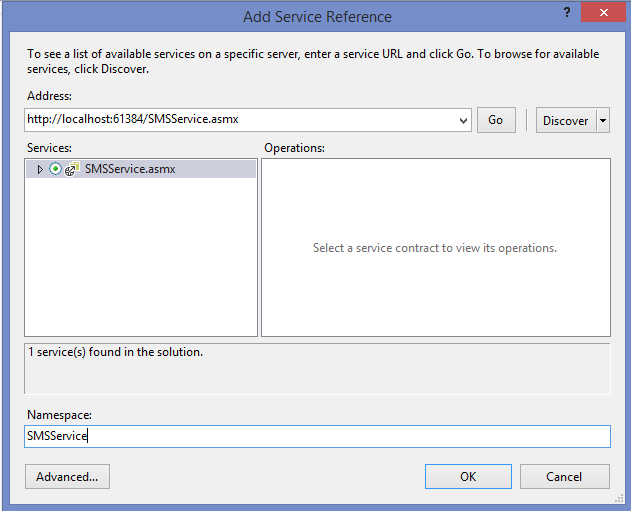
Open the project **ASP.NET Web Service** project - in the folder **SMSGatewayService**



1. Create a new project (Console, Windows, etc) and choose the target .Net Framework 4.0
2. Add service reference in your project



Enter the **address** of your ASP.NET Web Service, click **Go**



Press Ok and wait until service information is downloaded.

1. Copy paste the following code into your program

SMSService.SMSServiceSoapClient service = new SMSService.SMSServiceSoapClient();

service.Open();

service.SendSMS("0812", "sending message through ASP.NET Web Service");

service.Close();

# FAQ

This FAQ provides answers to basic questions about SMS Gateway.

## SMS Technology

* **What is SMS?**

SMS stands for Short Message Service. The term is also used to refer to a short message itself. It is a technology that enables the sending and receiving of messages between mobile phones in the mobile network.

* **Why use SMS?**

|  |  |
| --- | --- |
|  | SMS allows applications to directly transmit messages to mobile devices. It is the simplest way to send information. All that is needed is a telephone number and the message text. It is also convenient to receive SMS messages. You can collect information from mobile users this way. |

* **Is it possible to send a single message to multiple recipients?**

**Yes,** quite separate them with a semicolon.

* **Can I send longer text?**

**Yes,** SMS message can only carry a very limited amount of data (140 bytes). a technology called concatenated SMS (also known as long SMS or multipart SMS technology), makes it possible to send longer text or data.

Concatenated messages are supported by SMS Gateway.

* **What happen if an SMS Message is sent to a mobile phone that is offline?**

In standard SMS communication when an SMS is submitted by a mobile phone it is submitted to an SMSC. This message submission is called Mobile originated (MO) message submission. After the SMSC receives the message it stores it and looks for the recipient mobile phone. If the recipient mobile phone is available in the mobile network it sends the message as a Mobile terminated (MT) message.

## Example application of SMS Messaging

* **What applications can SMS messaging be used for?**

You can use SMS messaging for a lot of different purposes, include Person-to-Person Text Messaging, Information Provision, Notifications (E-mail, Fax and Voice Message Notifications, E-Commerce and Transaction Alerts), Two-Way Interactive Text Messaging Applications, and SMS Marketing.

* **SMS messaging for alert?**

**Yes,** you can use it for these purposes. You can use, for instance, SMS notifications in case of incoming e-mails. Whenever a new e-mail comes into your e-mail inbox, you can get an SMS notification of this. You can use this also for Fax and Voice messages. Since most people always have their mobile phones at hand, it is a good way to send information of important events and alerts, if needed.

## **SMS Gateway**

* **What is SMS Gateway?**

It is a software that makes it possible to send and receive SMS messages to and from the mobile network. This software offers various interfaces for the corporate office users, IT systems and corporate applications to send and receive SMS messages automatically or manually.

* **How does an SMS Gateway communicate with network?**

It directly through a wireless modem.

To connect an SMS Gateway to an SMSC you can use an SMS modem. An SMS modem is a standard mobile phone or a simplified mobile phone, that is called a GSM/GPRS modem. This mobile phone can be attach to the computer with a phone to pc data cable and it can be used to send and receive SMS messages.

* **What other functionality provided by SMS Gateway?**

An SMS gateway is responsible for handling capacity differences between the input and output channels. For example if an SMS application wants to send 10000 SMS messages at once, it will store these messages in a queue and will send them as capacity on the SMS service provider connection or GSM modem connection becomes available.

SMS routing is also built into most SMS Gateway's to make sure that incoming messages and outgoing messages end up where they should.

* **Do I have to pay for SMS if I have my own SMS Gateway?**

**Yes,** If you connect your SMS Gateway to the mobile network through a GSM or GPRS modem, you need a SIM card. The SIM card can be purchased from a mobile network operator. The SIM card will have an associated price plan, that will determine the cost of each SMS message.

* **Can I query a database to check incoming messages?**

**Yes.**

* **Can I send automatically reply messages to an incoming message?**

**Yes,** you can send automatic reply using Plugin. It also possible to query the reply based on a keyword of the incoming message.

## **Hardware and Software requirements**

* **What is the minimum hardware requirement for the PC I will use to run SMS Gateway?**

**CPU**: Intel Pentium III, 800 Mhz  
**RAM:** 512 Mb  
**Hard disk space:** 50 Mb  
**Ports**: RS232 and/or USB (if you use a GSM modem)  
**Network:** if you use IP SMS connection

* **What is recommended hardware for PC I will use to run SMS Gateway?**

**CPU:** Intel P4D, 3Ghz  
**RAM:** 2 GB **Hard disk space:** 500 Mb (SATA or SCSI) **Ports:** RS232 and/or USB (if you use a GSM modem) **Network:** Internet connection (if you use IP SMS connection)

* **What is the recommended operating system?**

**Windows 8**

**Windows 7**

**Windows Vista**

* **Does SMS Gateway support 64bit operating system?**

**Yes.**

* **What are software requirements?**

**OS: Windows XP SP2, Windows 2003, Windows Vista or Windows 7**

**Installed software: .NET Framework 4.0**

* **Do I need dedicated server?**

**No.** you do not, The SMS Gateway can be used with other applications installed on the same hardware.

* **Will the software run on Virutal Machine?**

**Yes,** it will.

If you use IP SMS connection, the SMS Gateway will work well. If you use an USB GSM modem you might experience problems getting the modem to work in your Virtual Machine. You must make sure the driver that comes with the USB modem supports the Virtual Machine.

* **Which GSM device do you recommended?**

**Wavecom**

To choose a working device, please make sure the phone has the following features listed in its technical specifications:  
  
- The phone should have a built-in modem.  
- The phone should have a USB or RS232 data cable.  
- The phone should support the GSM 7.05 standard for SMS messaging.

If you use smartphone as GSM device like Samsung, the sms gateway functions will not work properly and may only be able to send sms.