QuesCom SMS Solutions

Table of content

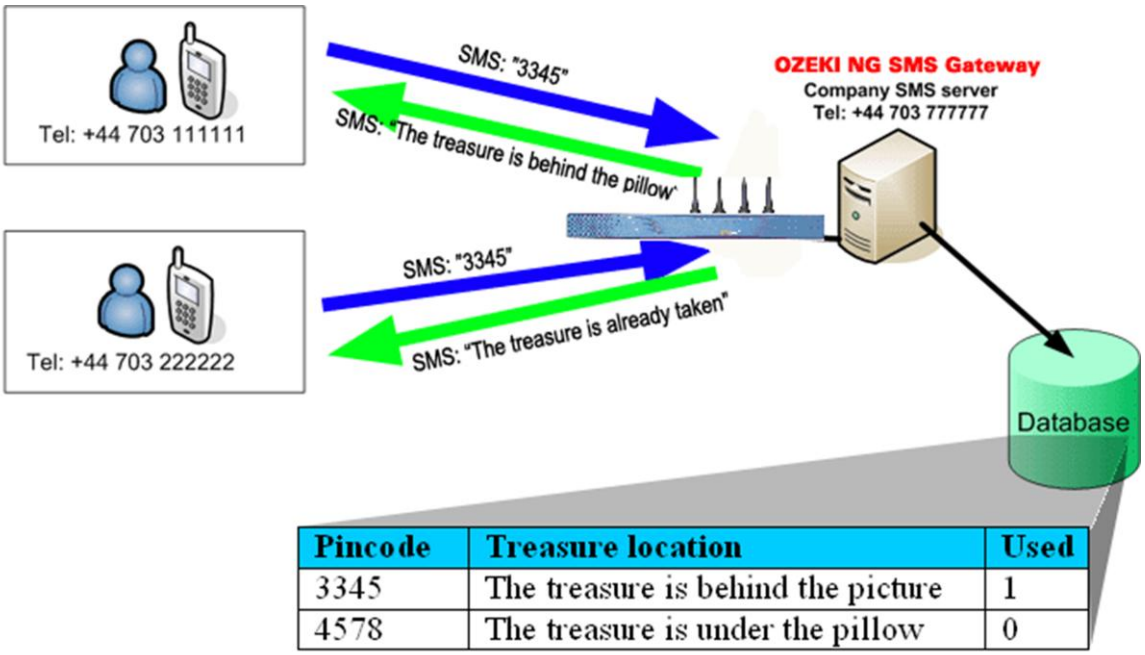
www.quescom.com

1. SMS PIN code game

This guide shows you how to create an example SMS application and gives you the steps to create it. It uses Ozeki NG SMS gateway platform, a QuesCom GSM gateway and an ASP script to process incoming SMS messages. You may use and modify this ASP script freely to accommodate your needs.

Introduction

The application is PIN code game. The users of the game can send in PIN codes to find out treasure locations. If the PIN code matches an entry in the database, the appropriate treasure location is returned. A treasure location is only returned to the first person who sends in the correct PIN code, the other who are late will receive the "Treasure is already taken" message (Figure 1).



Pincode	Treasure location	Used
3345	The treasure is behind the picture	1
4578	The treasure is under the pillow	0

Figure 1 - Application layout

How to setup this solution

After you are able to send and receive SMS messages, you need to create a database that will hold the data for the game. Any kind of database can be used. In our example we use Microsoft SQL Express. (It can be downloaded free of charge from Microsoft.)

The next step is to install an ASP User in Ozeki NG SMS gateway. This will allow you to execute an ASP script when an SMS comes in. The ASP script will hold the application logic, that will process the SMS message and will generate the response SMS messages.

The source code of the ASP script should be saved into the following directory to get this solution working:

```
C:\Program Files\Ozeki\OzekiNG - SMS
   Gateway\config\ASPDemo\sample.aspx
```

The source script can be downloaded from the http://www.ozekisms.com/ website: sample.zip (2 KB)

How can I test this solution?

The system can be tested easily. You can send an SMS message to the QuesCom GSM gateway. After the message is sent, you will receive a response SMS. Of course you will probably want to modify the ASP script, that does the processing according to your needs. If you modify the ASP script, you will find it useful during the development to see the raw script output. The script output that was generated when an SMS arrived to the gateway can be found at the following location: "C:\Program Files\Ozeki\OzekiNG - SMS Gateway\config\ASPDemo\output.htm";

2. SMS Order System

This case study gives you an example on how you can use SMS messages to keep track of the stock of goods in your shops. It explains the simple idea to collect order information in SMS messages and to save this information in a database. The database can be used to do reports or automate business processes. The case study starts out by explaining the scenario, then it gives a step by step explanation on how to setup this system.

Introduction

The example presented here can be used to keep track of orders arriving from cafe shops around the country. In our example there are three cafe shops, one in Liverpool (tel: +44 703 111111), one in Glasgow (tel: +44 703 222222) and one in London (tel: +44 703 333333). Each of these cafe shops has a cellphone number that identifies the shop itself. If the stocks are running low in the shop, the shop owner can use the mobile phone to send an order in an SMS message (Figure 1). The order is received by the company SMS system. The SMS system processes the order and saves the data into a database.

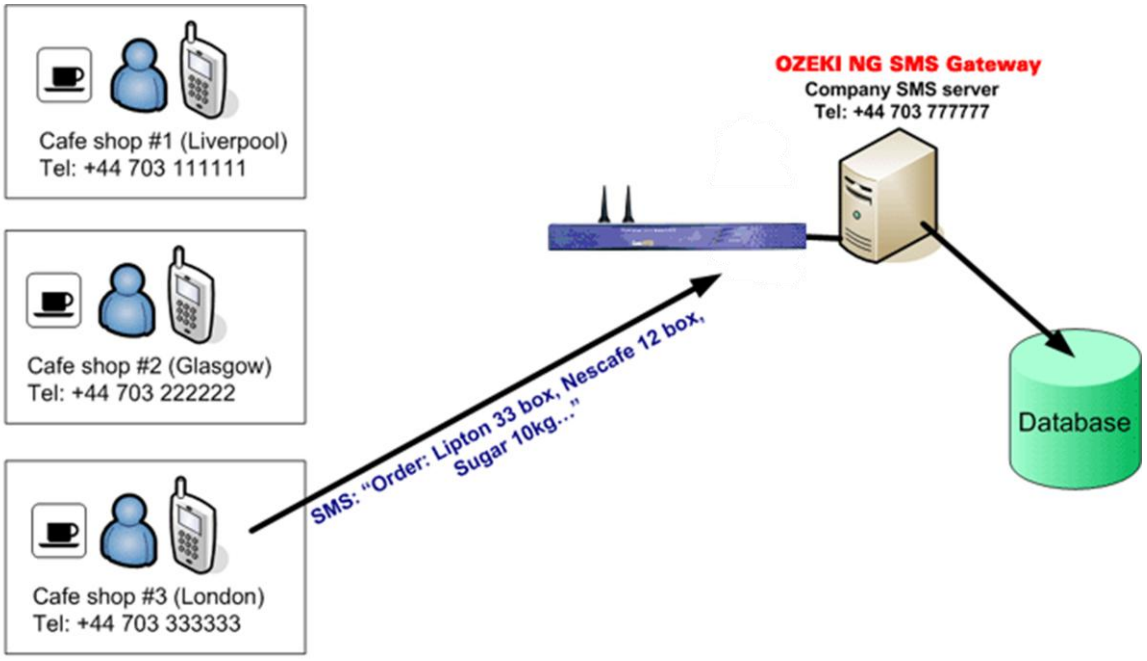
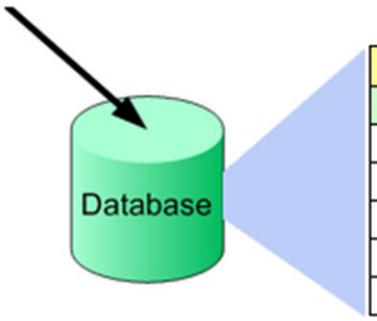


Figure 1 - Usage scenario for SMS order system

The database layout for the order contains the shop telephone number, the time of the order and the quantities of the ordered products (Figure 2). This is a very simple layout to keep this example easy to understand. Of course you can change the database layout to accommodate your needs.



Orders				
Shop	OrderDate	Nescafe	Lipton	Sugar
+44 703 111111	12/01/2008	3 box	-	-
+44 703 333333	12/01/2008	2 box	4 box	3 kg
+44 703 333333	14/01/2008	33 box	12 box	10 kg

Figure 2 - Database layout for SMS order system

The database table containing the orders can be stored in any database server, such as Oracle, Microsoft SQL Server, Access, Sybase, MySQL or other. You can even use Excel files.

Prerequisites

To setup this system you need the following:

A PC with Windows XP, 2003 or Vista	We recommend to use a standard PC with Windows XP. The recommended hardware configuration can be found in the prerequisites section of our SMS Gateway product manual.
QuesCom GSM Gateway	Any of the QuesCom 200 or QuesCom 400 GSM gateway.
Ozeki NG SMS Gateway	This software should be installed onto your PC. It will handle your QuesCom GSM gateway (or your IP SMS connection) and will do the SMS receiving. It can be downloaded from the following url: download SMS Gateway.
Database server	You can use Oracle, Microsoft SQL Server, Access, Sybase, MySQL or other. In the example we use Microsoft SQL Express, because it can be downloaded free of charge from Microsoft.

How does it work?

Once you have all the prerequisites in place the system will work the following way: The SMS message will be received by your QuesCom GSM gateway (Figure 3). Ozeki NG SMS Gateway will read the incoming SMS from the QuesCom GSM gateway and will pass it to an ASP script. The ASP script will process the message. During the processing the ASP script will take the text of the SMS messages and will put the data in it into the database server.

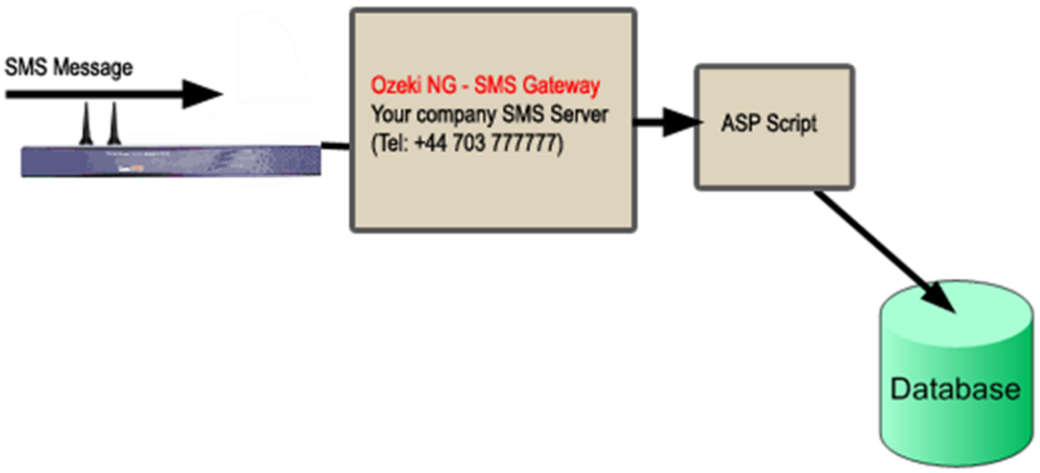


Figure 3 - System architecture for SMS order system

Installation steps

To get this solution working you need to follow these steps. If everything goes well, you can get the system working in about 10 minutes. I assumed you have already downloaded and installed the latest version of Ozeki NG SMS Gateway. The steps here give you information on how you should configure Ozeki NG SMS Gateway to get your solution working.

Step 1 - Install and test your QuesCom GSM gateway. Setup QuesCom GSM gateway and Ozeki link (new page to be created, content at the end of that document)

Step 2 - Create your database layout In our example we use Microsoft SQL Express as our database servers. The database layout (Figure 4) consists of only a single database table. This database table can be created using an SQL CREATE TABLE definition in an SQL console. Detailed steps for creating the database table layout can be found in the SMS Order System Database Layout page.

```
CREATE TABLE orders (
```

```
 id int IDENTITY (1,1),
 shop varchar(30),
 orderdate varchar(160),
 nescafe varchar(160),
 lipton varchar(160),
 sugar varchar(160)
);
```

Figure 4 - Database table layout for SMS Order system

Step 3 - Configure the ASP script - The ASP script will process incoming SMS messages and will save data into the database. The script will be executed by Ozeki NG SMS Gateway whenever a new SMS comes in. To configure Ozeki NG SMS Gateway to execute this script you need to add an "ASP User" to the Ozeki NG configuration. This can be done by clicking on the "Add" menu item in the "Users and applications" menu, then you should click on "install" at the "ASP" section (Figure 5).

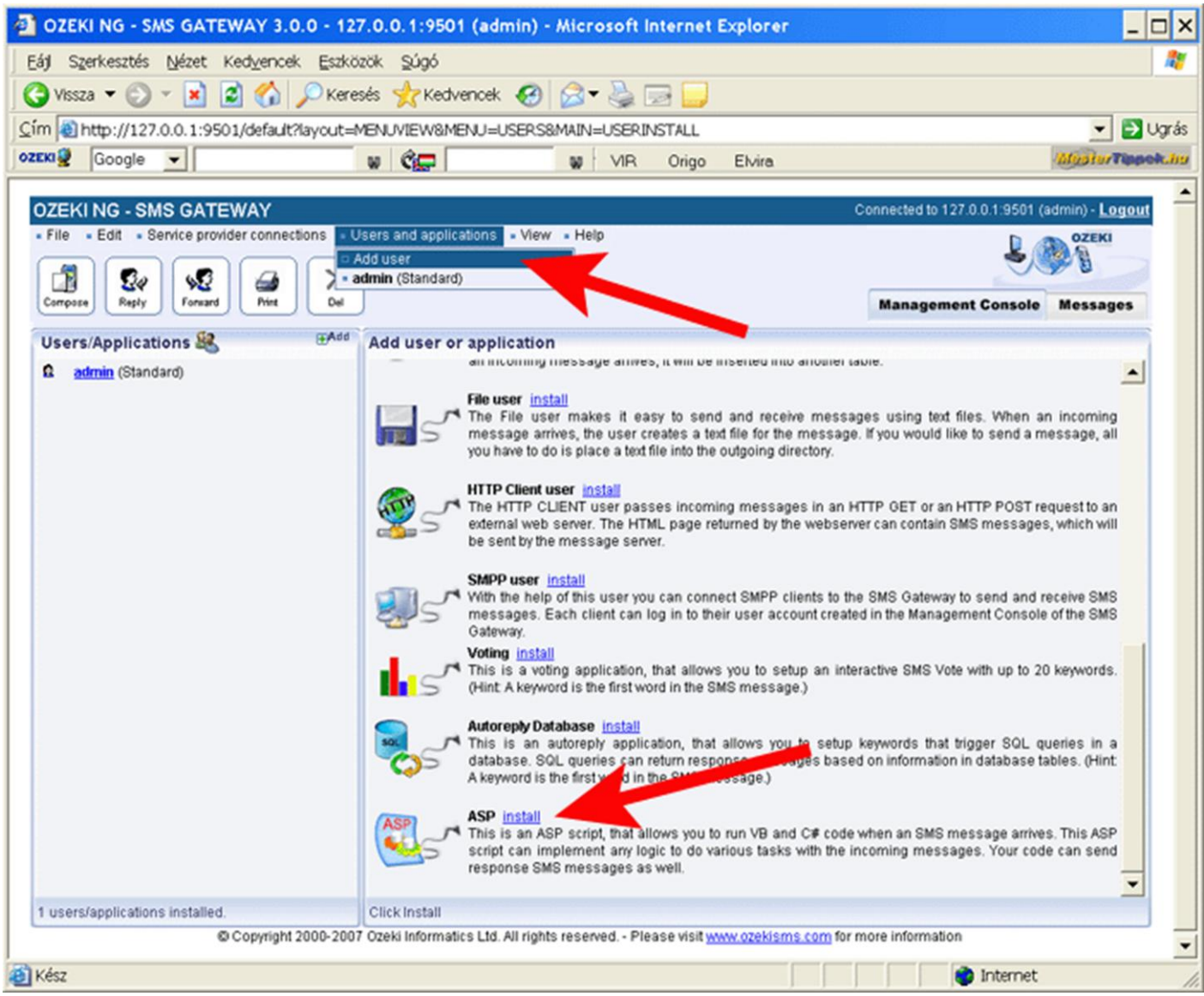


Figure 5 - How to add an ASP SMS script to the SMS gateway

After the "ASP" application installation is started, you have to supply a name for the application. You can pick any name. In the example we use "ordersystem" (Figure 6).

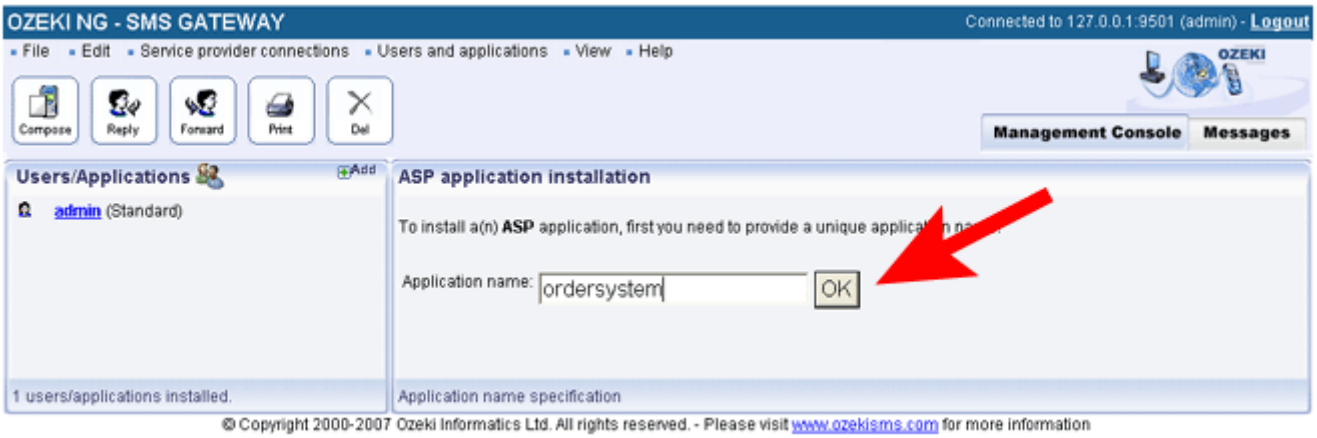


Figure 6 - How to set the ASP SMS application name

Step 4 - Create the ASP script - The final step is to edit the ASP script to implement the application logic, that will process the incoming SMS message. The ASP script that should be edited can be found at

C:\Program Files\Ozeki\OzekiNG - SMS Gateway\config\ordersystem\sample.aspx

you can edit this script with notepad or any other text editor. The source code of the script can be found in the sample ASP script for SMS Order System section.

How can I test the system?

The system can be tested easily. You can send an SMS message to the phone number of the QuesCom GSM Gateway that is attached to your computer. After the message is sent, you should check your database table and hopefully the processed information from the received SMS messages will be there. Of course you will probably want to modify the ASP script, that does the processing according to your needs. If you modify the ASP script, you will find it useful during the development to see the raw script output. The script output that was generated when an SMS comes in can be found at the following location: "C:\Program Files\Ozeki\OzekiNG - SMS Gateway\config\ordersystem\output.htm";

Conclusion

The example presented in this chapter introduces a nice idea to use SMS messages to collect information from the field. It shows how you can use ASP to process the incoming information and save it in a database. It is useful if you wish to create similar solutions.

3. SQL SMS Gateway

If you use the Ozeki NG - SMS Gateway, you can send and receive SMS messages using a database server with the help of SQL queries. In order to use this option, you need to have a database server (such as Oracle, Access, MySql, MS SQL, Postgres, etc.) installed, and two database tables should be created: ozekimessageout and ozekimessagein. One of these will be used for sending and the other for receiving SMS messages.

The SQL SMS Gateway is excellently convenient for IT professionals to enhance established IT systems by adding SMS functionality to web-based applications (ASP, PHP) and enterprise management software. The integration of the SQL SMS Gateway will enable you to send and receive a large amount of SMS messages and to establish automated SMS notification and processing systems.

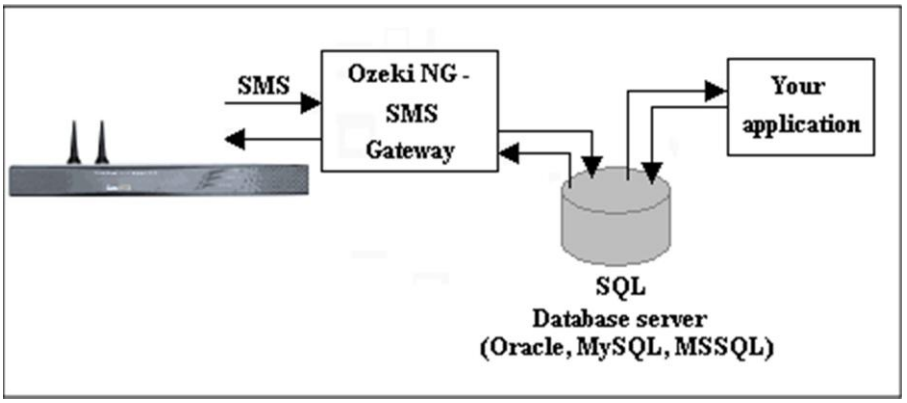


Figure 1. - OZEKI NG - SMS Gateway connecting to a database server

Ozeki NG - SMS Gateway will connect to the database through a standard ADO or ODBC connection (Figure 1). Using this connection, it will periodically query the ozekimessageout database table for outgoing messages using a SELECT statement and will INSERT incoming messages into the database table used for incoming messages (ozekimessagein).

Setup QuesCom GSM gateway and Ozeki link (new page to be created, content at the end of that document)

4. SMS content query with PIN code

This is a sample applications for SMS based content delivery with PIN code authentication. The application works the following way: The mobile users sends in an SMS message with a pin code. The software checks to see if the sender phone number and the pin code is in the database. If they are in the database the software returns the appropriate response message.

This solution can be created with the "SMS Autoreply from Database" solution presented at http://www.ozekisms.com/index.php?ow_page_number=392

To get this solution working, you need to create this database table in your SQL server:

```
CREATE TABLE content ( id int
 IDENTITY (1,1), senderphone
 varchar(30), pincode
 varchar(16), responsemessage
 varchar(160),
);
```

And you have to put the following line into your Autoreply database user script.

```
n.*
SELECT '$sender',response message from content where keyword='$keyword'
```

5. SMS newsgroup / SMS opt-in list

In this example we show you how to setup an SMS newsgroup. An SMS newsgroup is an opt-in list. In this example, people can join the list by sending in the word "join" in an SMS message and can leave the list by sending in the word "leave". If somebody sends in an SMS message starting with the keyword "news", all the members of the list will receive the SMS.

Preparation

Download Ozeki NG SMS Gateway and to install it. Installation is similar to the Installation of any windows application. Ozeki NG SMS Gateway will control your QuesCom GSM Gateway and will make it possible to create an SMS news group. After Ozeki NG has been installed, the QuesCom Gateway should be setup. The final step is to test the functionality of your system, by sending and receiving a test message.

Overview

The SMS newsgroup we are going to create will work the following way: The first user with phone number +4411111 subscribes to the list by sending in the work "join". The system will automatically respond with a reply message (Figure 1).

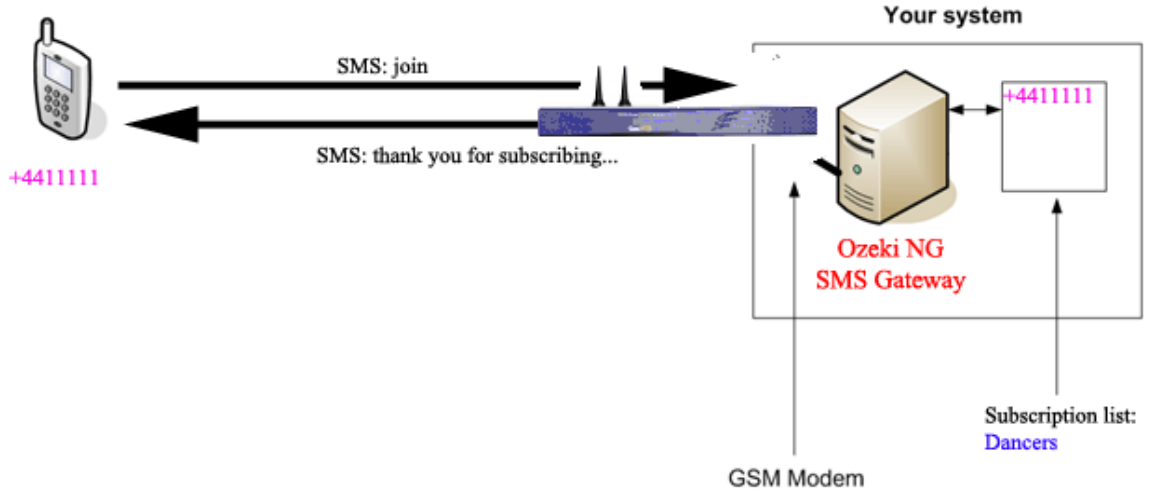


Figure 1 - The first user subscribes to the list

The second user with phone number +4422222 subscribes to the list the same way (Figure 2).

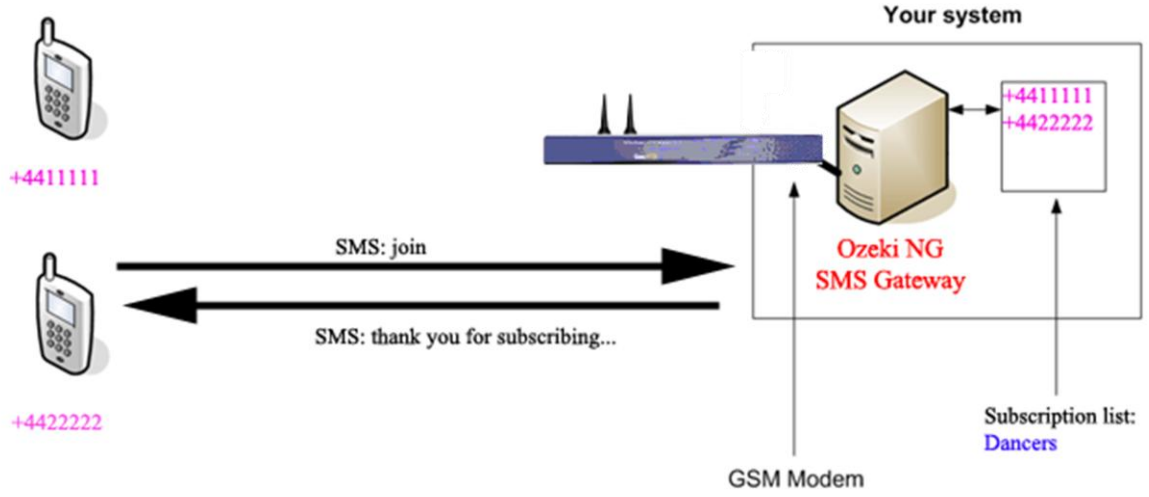


Figure 2 - The second user subscribes to the list

The third user with phone number +4433333 subscribes to the list the same way (Figure 3).

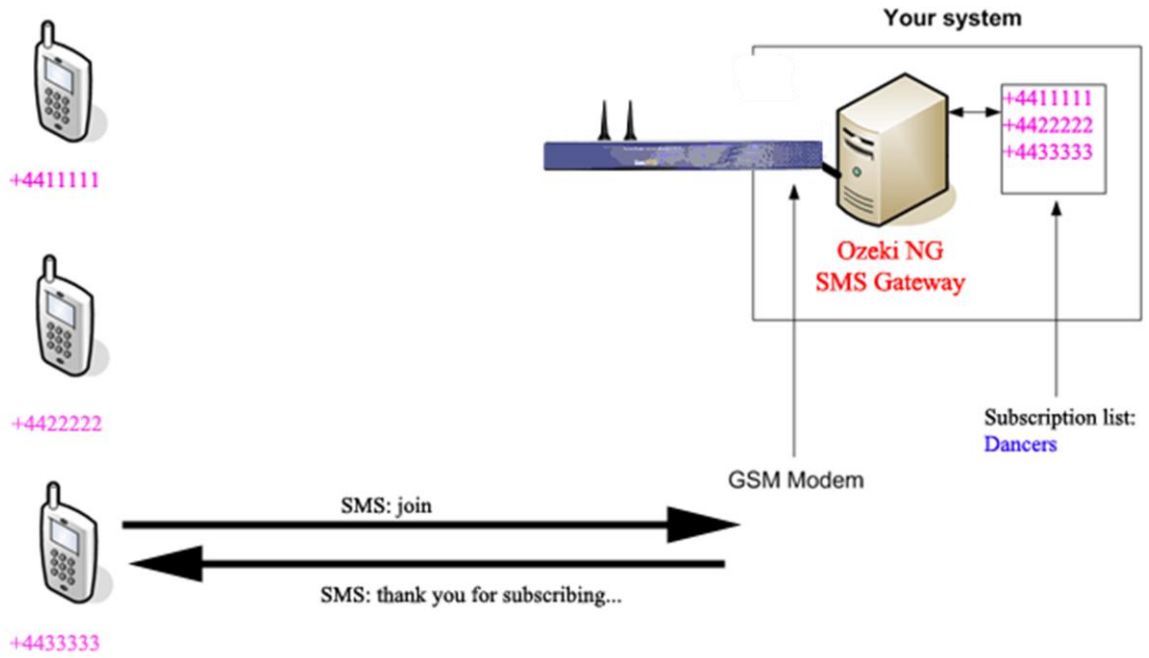


Figure 3 - The third user subscribes to the list

After everybody is subscribed, the first user with phone number +4411111 sends a message to the list. The message is broadcasted because it starts with the keyword "news". This means the the second user with number +4422222 and the third user with number +4433333 will receive a copy of the message (Figure 4).

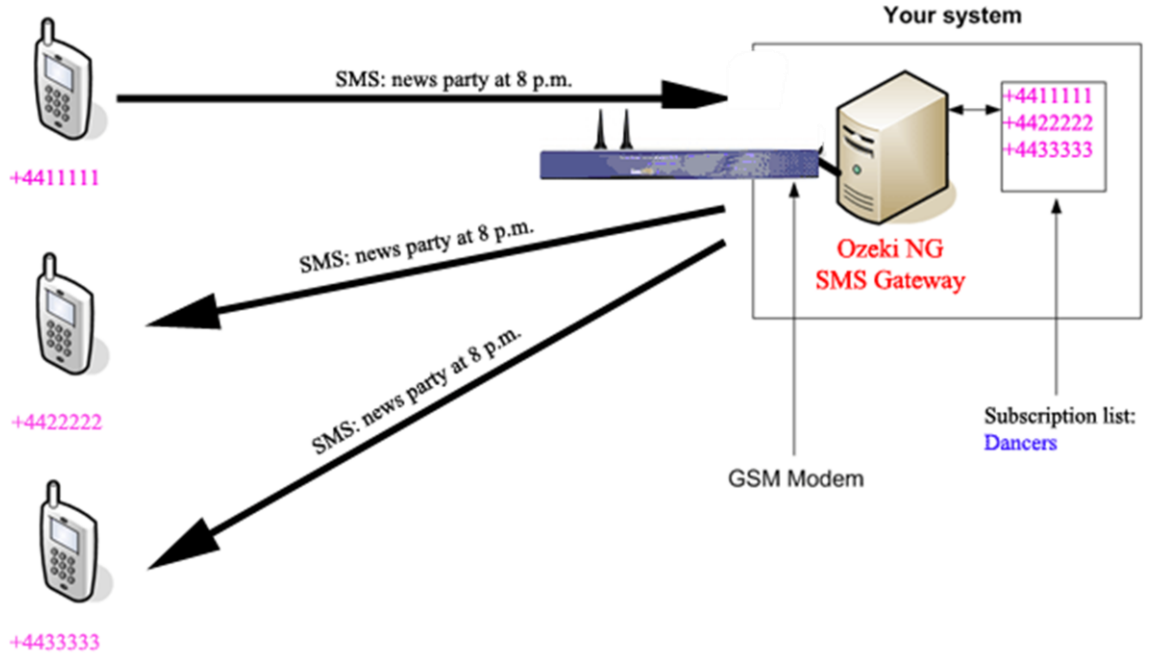


Figure 4 - The first user send a message to the list.

How to configure

Then you have to log into Ozeki NG SMS gateway as administrator (Figure 5). The default username and password is admin/abc123).

Figure 5 - Login to the SMS Gateway as admin

Then you have to add an autoreply user. This can be done by clicking on the "Add new user link" (Figure 6), and clicking on "install" in the Add user or Application list (Figure 7).

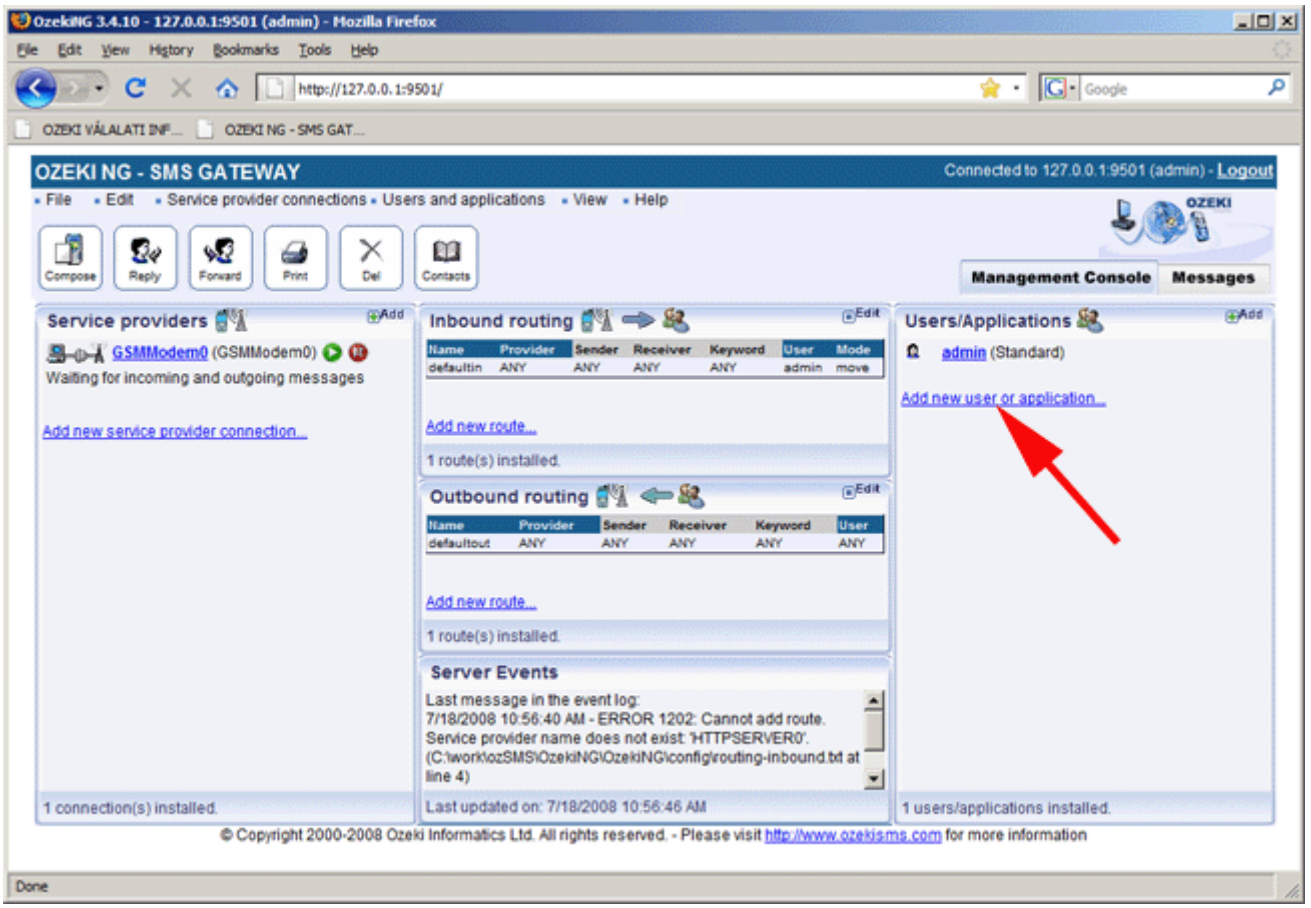


Figure 6 - Add new user

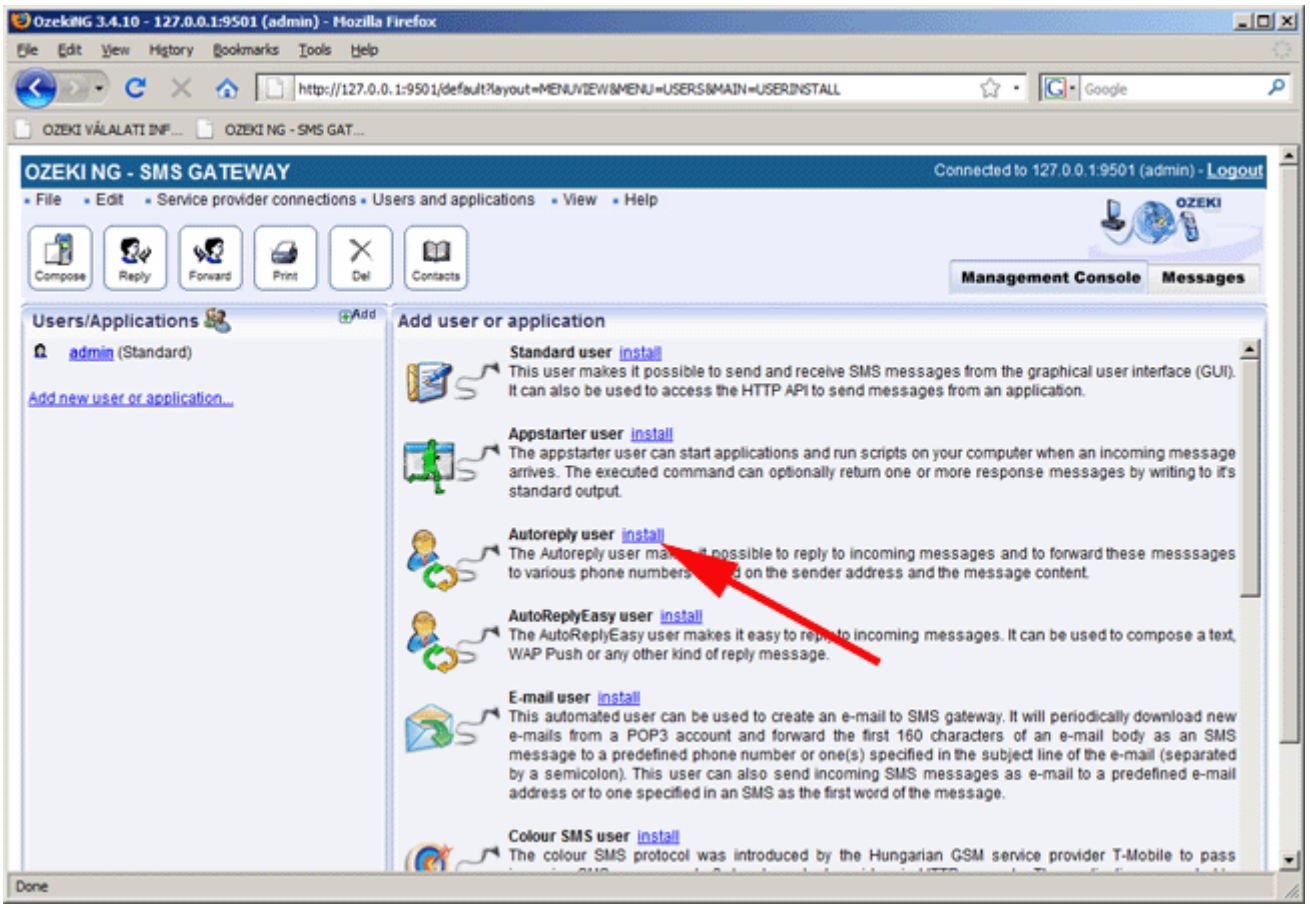


Figure 7 - Select the Autoreply user

Once the Autoreply user is added, you need to configure it. During the configuration first you have to add a unique name, such as "danceclub" (Figure 8). The next step is to create a script that will send all messages that start with the word "news" to the members of the group called "dancers" (Figure 9).

Here is the script:

```
m^news.*
grp://dancers MSG
```

The script is very simple. It contains two lines. The first line matches the keyword in them message, the second line will forward the message text to the group called "dancers". Note that the dancers group does not exist at this point. It will be created in the next steps. Finally you need to enter a password for interactive login (Figure 10).

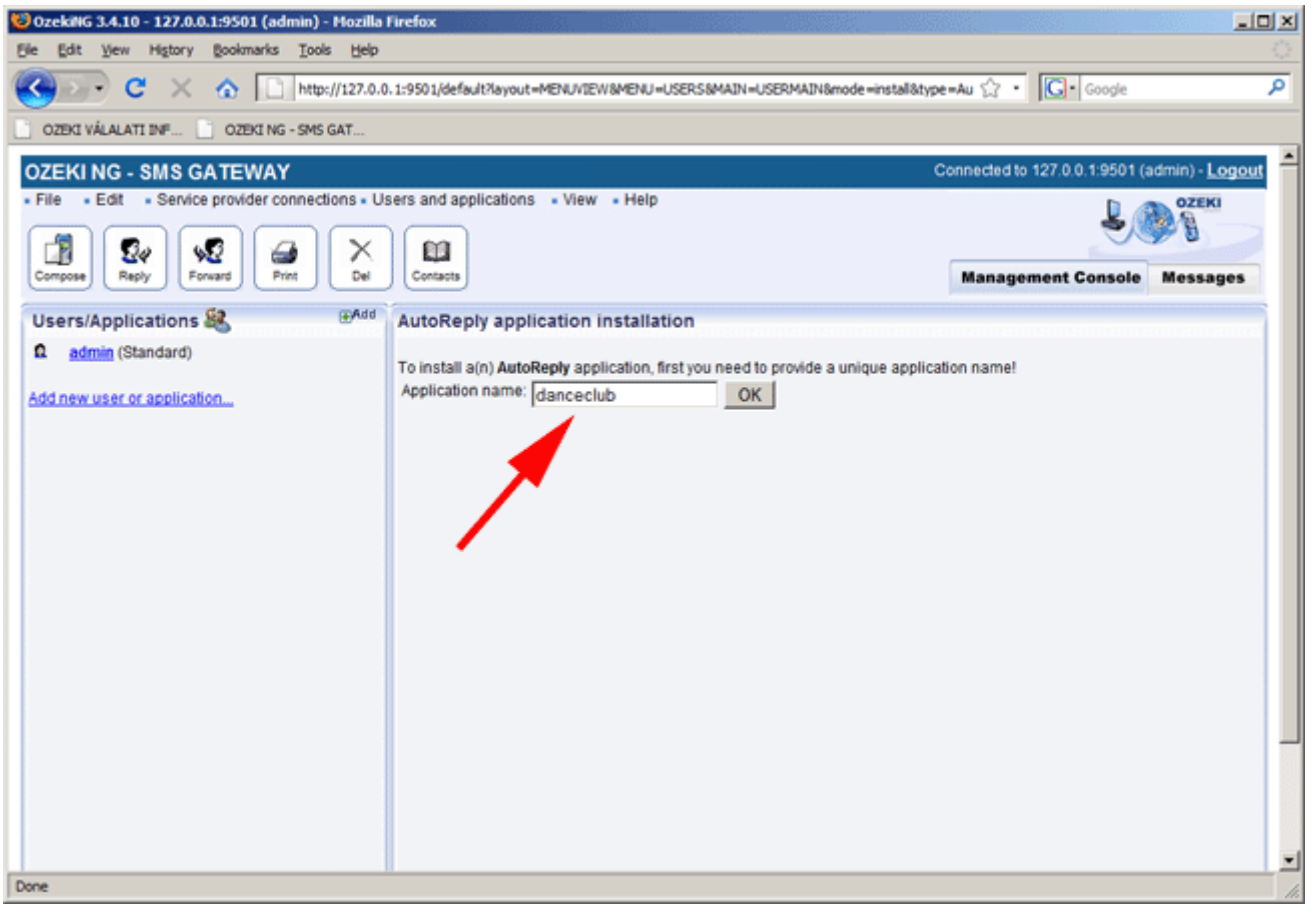


Figure 8 - Create a name for the service

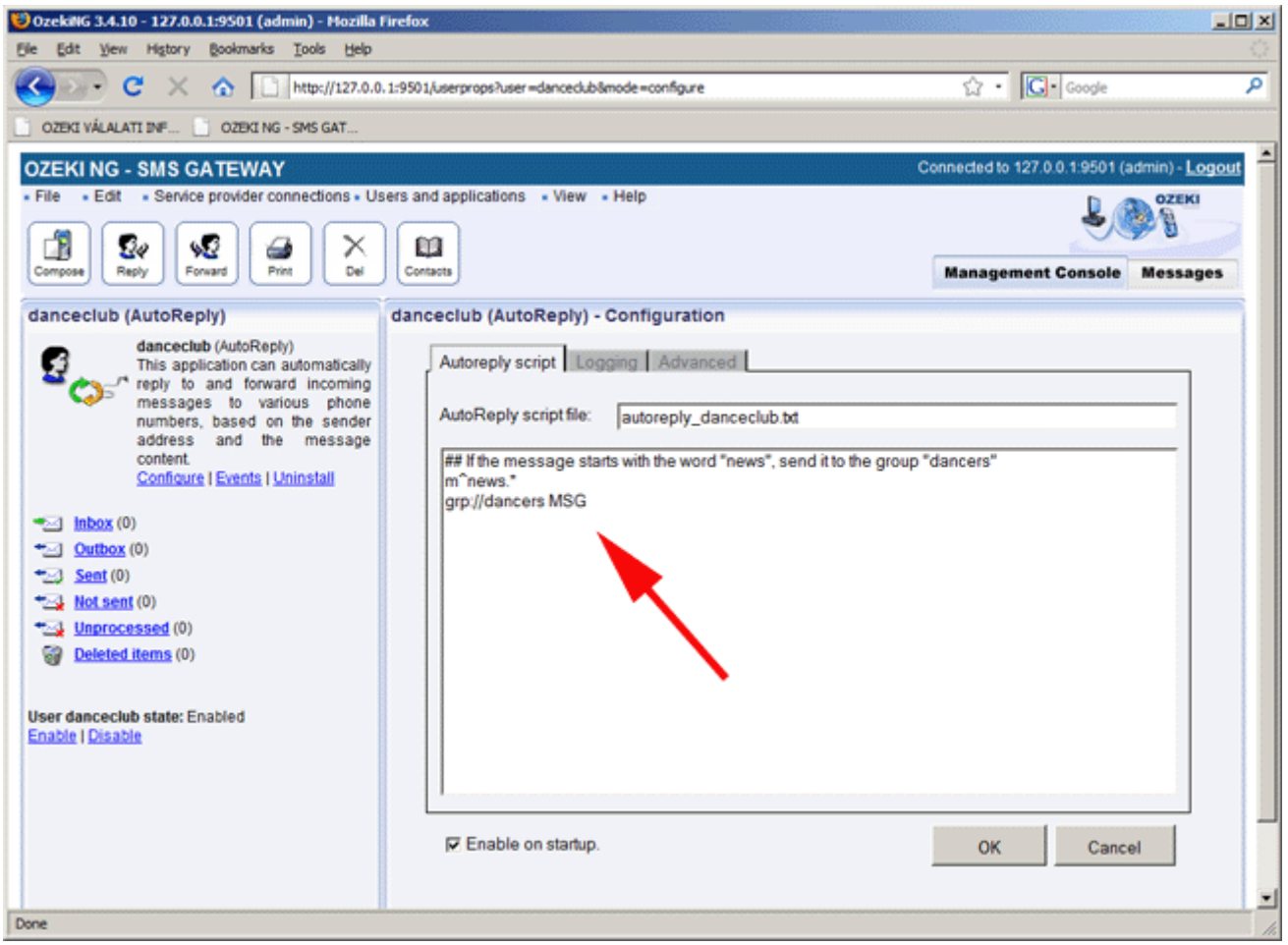


Figure 9 - Compose the SMS autoreply script

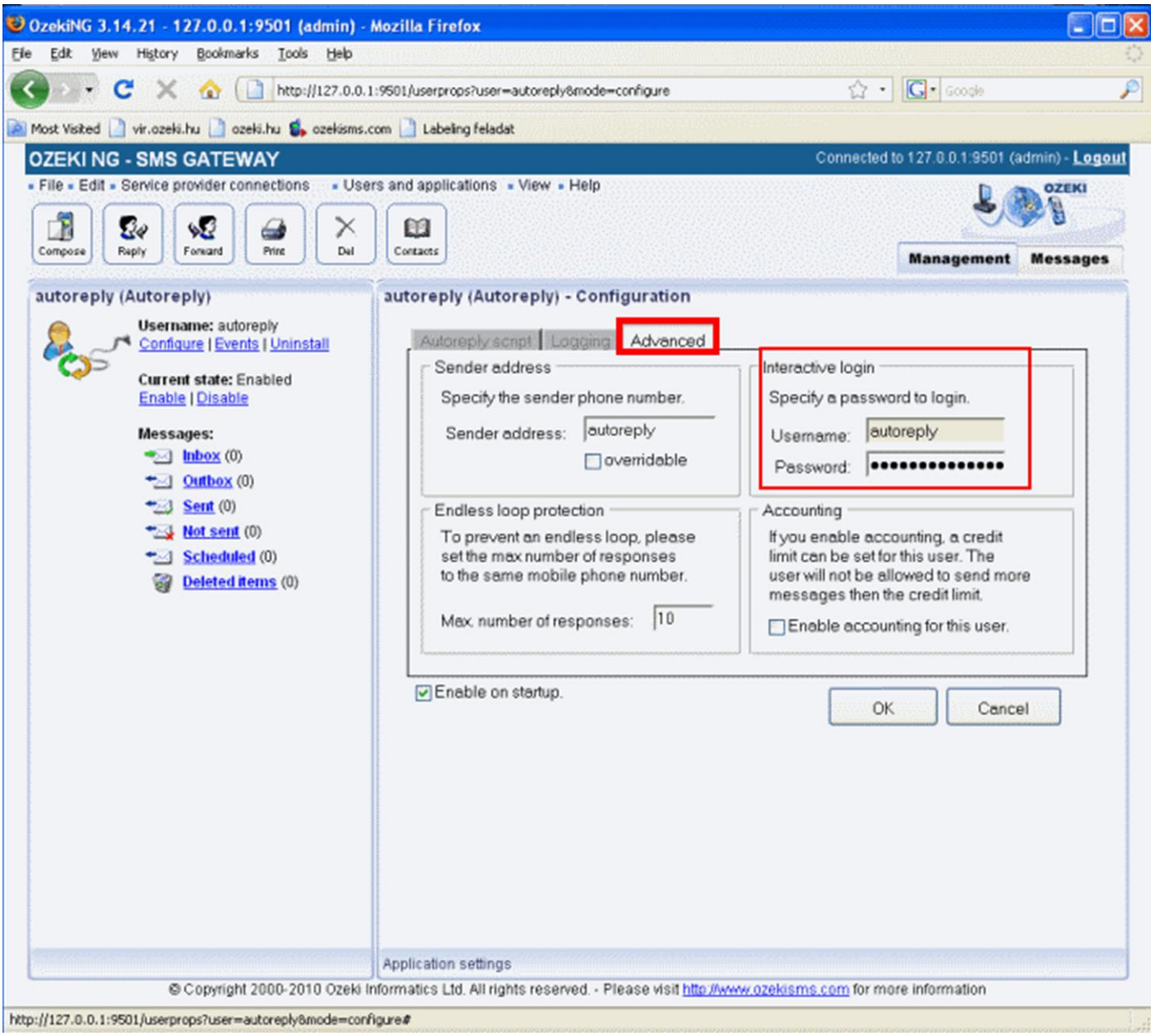


Figure 10 - Create a username and a password

After the you have configured the autoreply user, you need to logout (Figure 11) and log back in as the autoreply user (Figure 12). To log back in (Figure 13) you can use the name of the autoreply user and the password you have specified at the Interactive login section of the autoreply user configuration form.

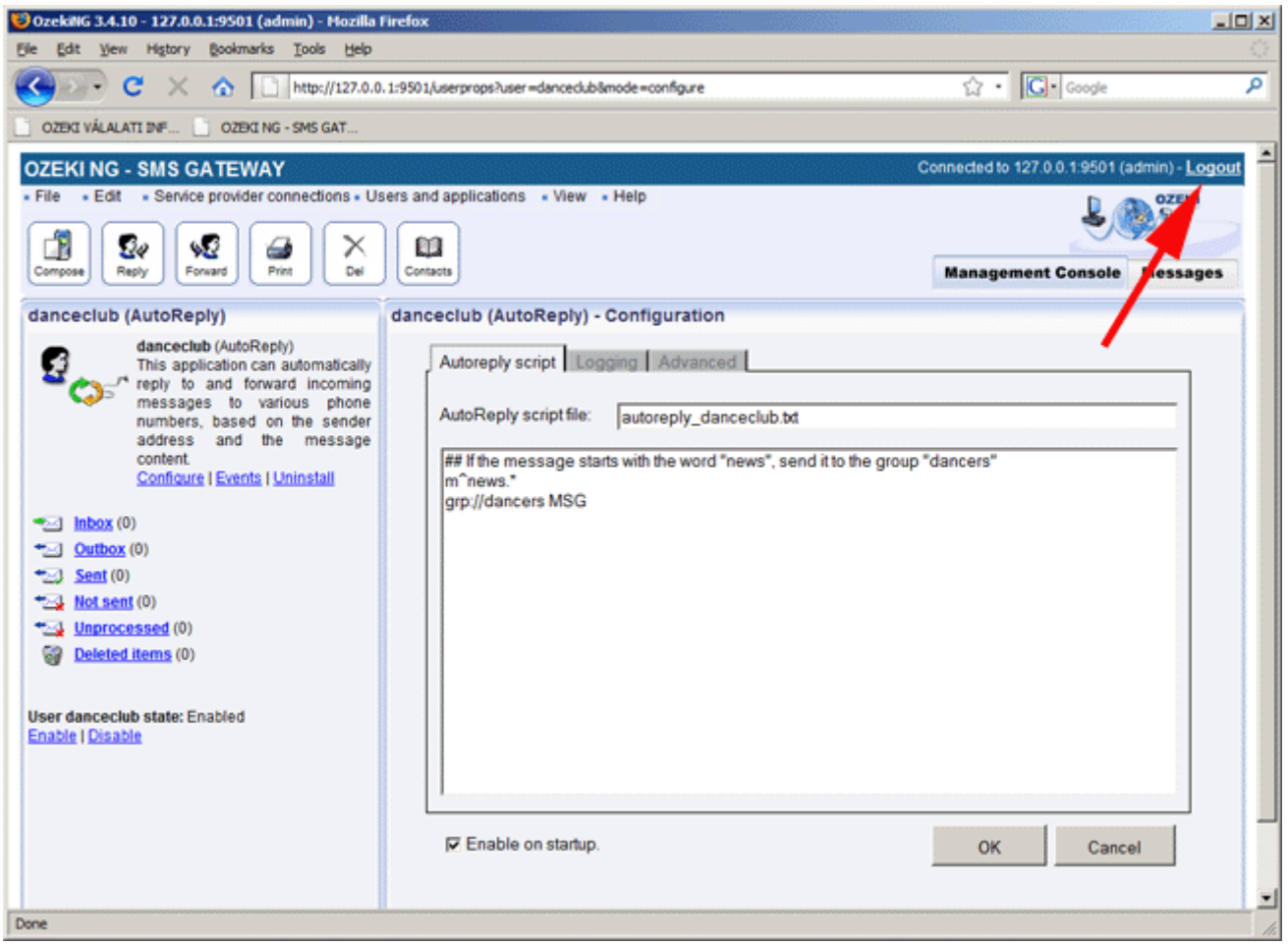


Figure 12 - Logout from the admin account

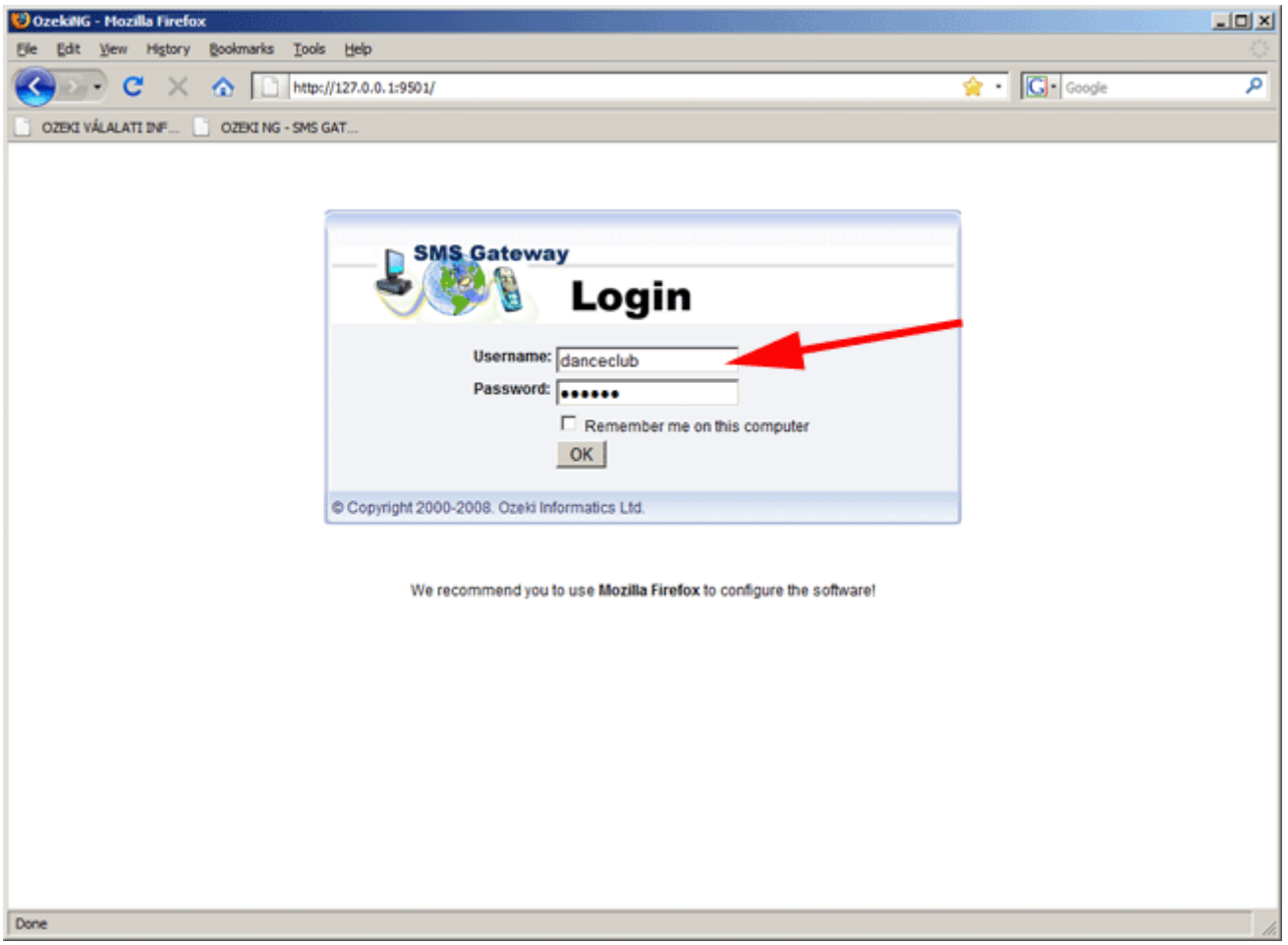


Figure 13 - Login as the autoreply user

After you have logged in as "danceclub", you should open the addressbook by clicking on the "Contacts" icon (Figure 14). In the addressbook you need to create a new group (Figure 15) called "dancers". Note that the name "dancers" matches the name used in the autoreply script. When you create the new group (Figure 17), in the configuration form, open the Subscribe and Unsubscribe tab. In this tab you can specify the subscribe and unsubscribe keywords along with the greeting messages. In the example we have specified "join" as the subscribe keyword (Figure 17).

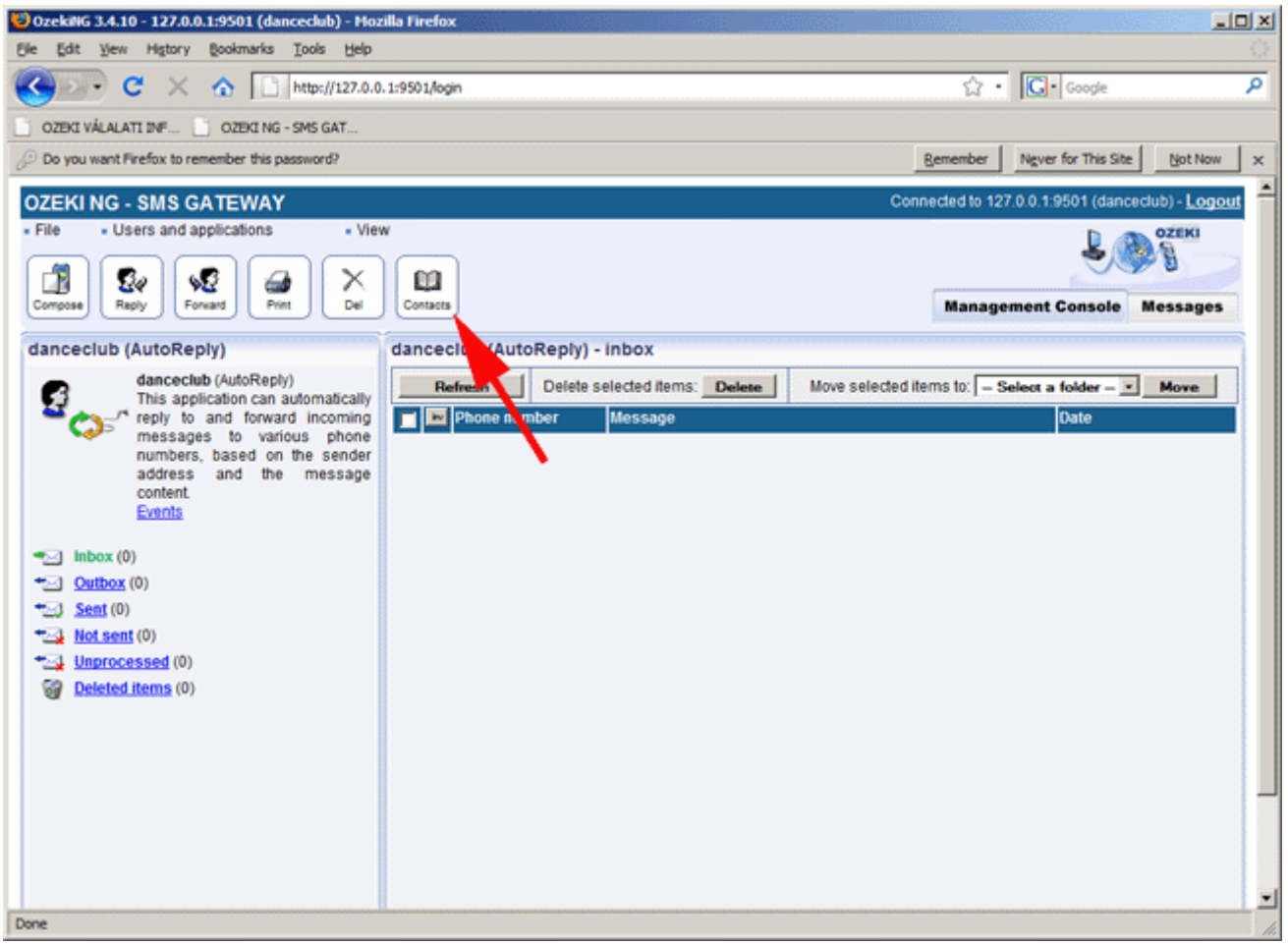


Figure 14 - Open the address book

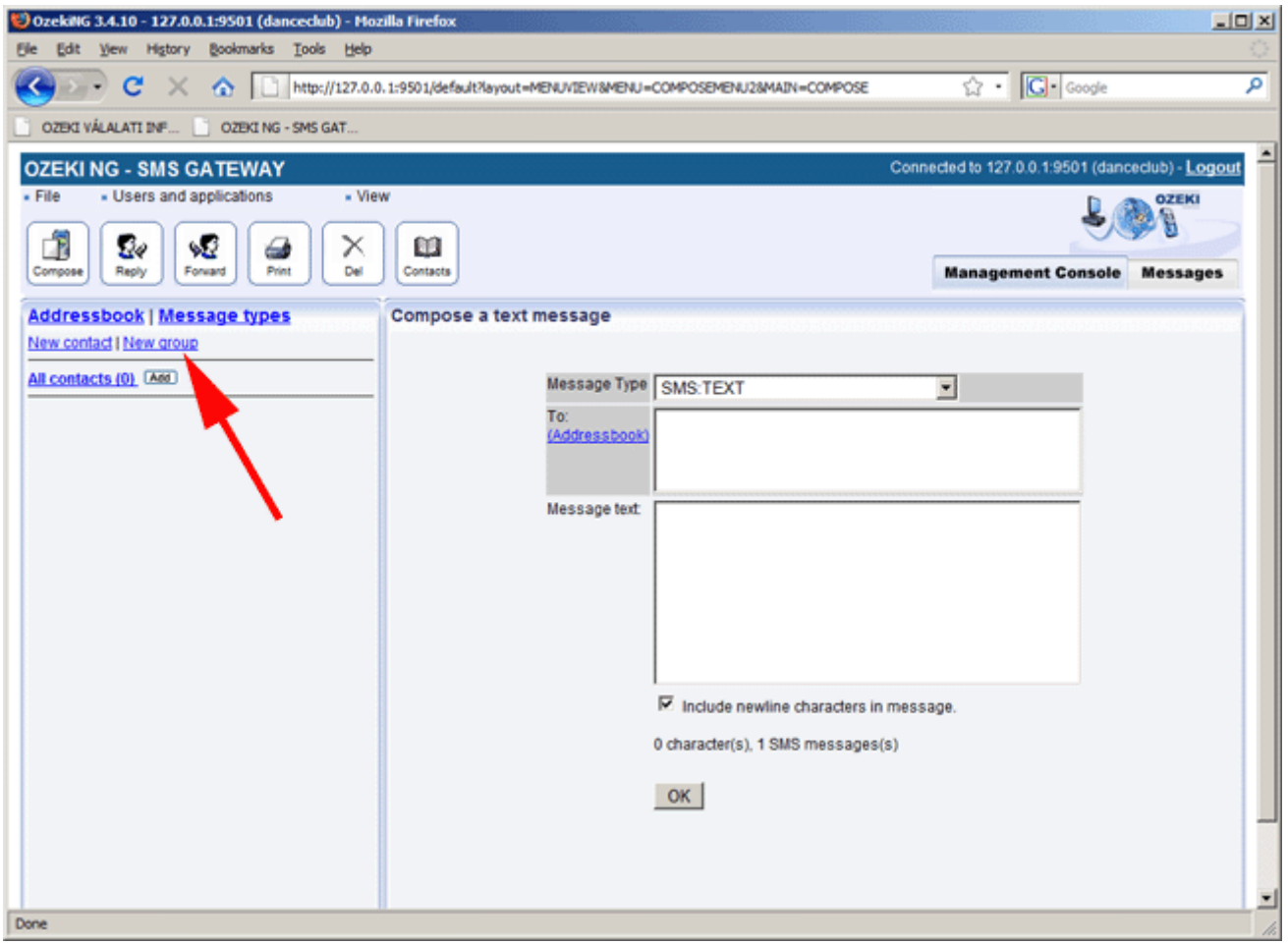


Figure 15 - Create a new group called "dancers"

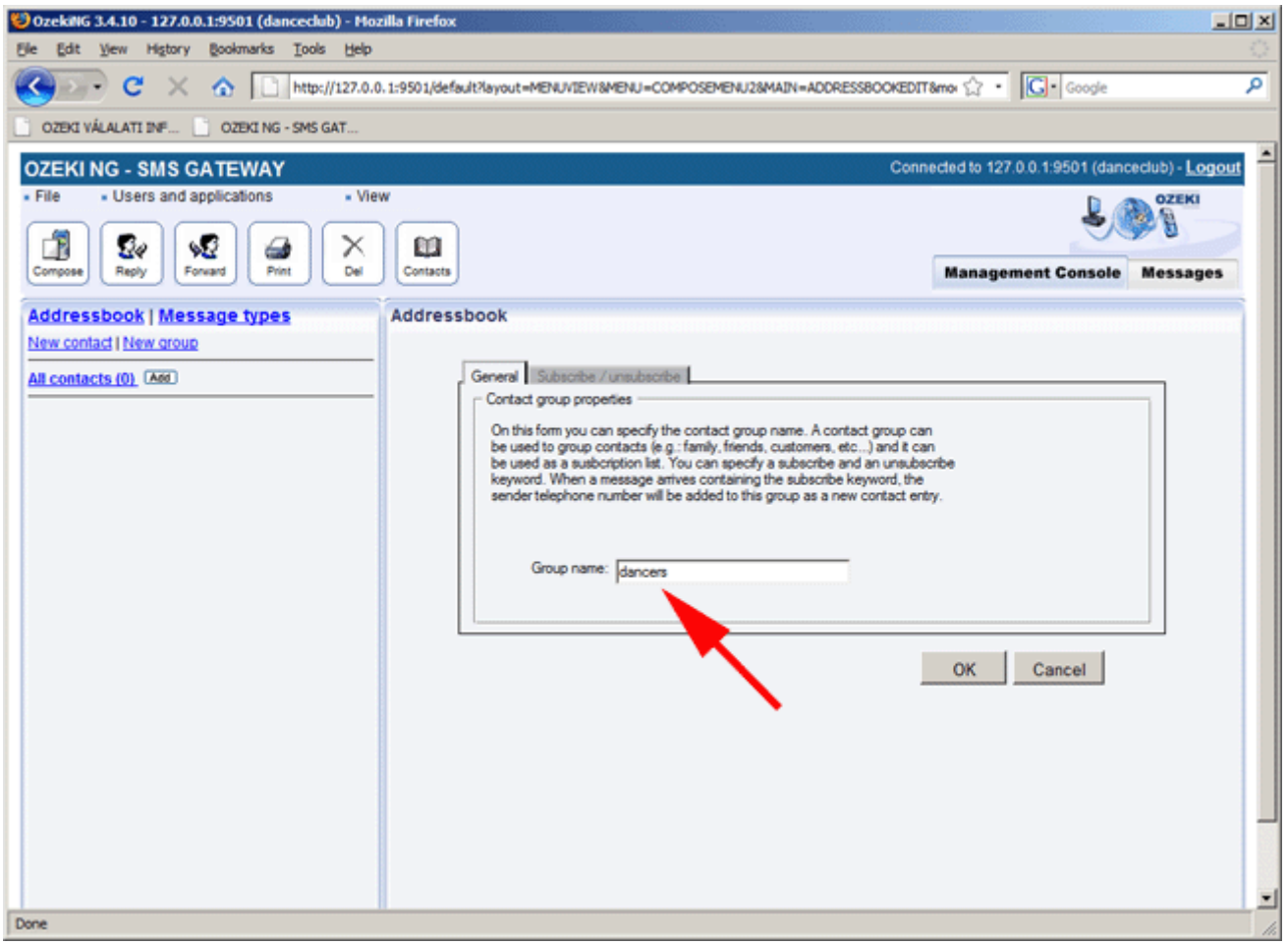


Figure 16 - Specify the group name

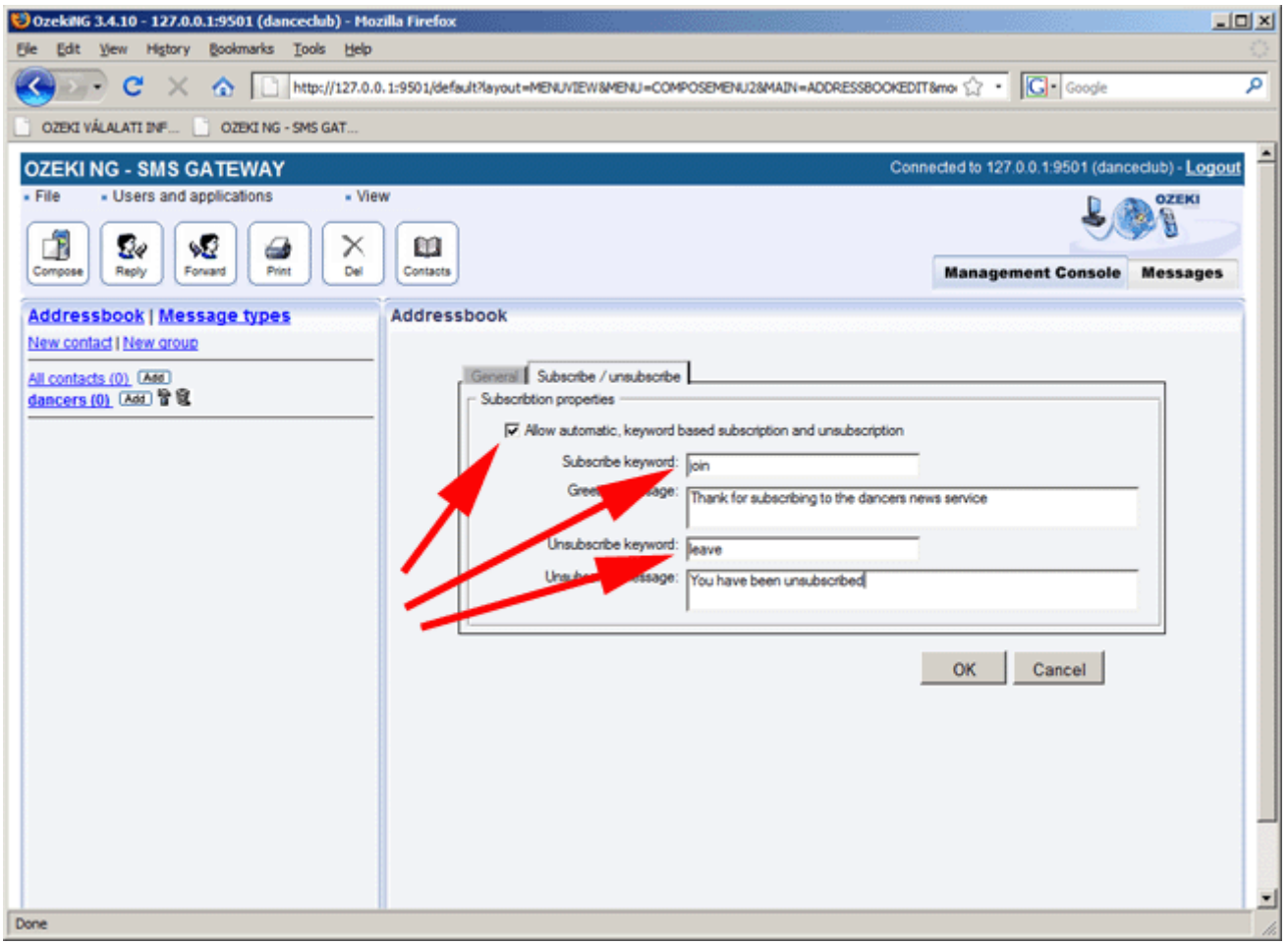


Figure 17 - Specify subscription settings

Finally you need to click OK and you need to log out from the autoreply user account. You can test the system, by sending in the SMS message "join".

6. Information service menu via SMS

This example shows you how you can use Ozeki NG SMS Gateway to create an SMS information menu. The SMS information menu is based on an automatic response system and works similarly as games and other service menus. It works very simply, the mobile users send messages including certain characters (numbers or letters) to a specified phone number and the system replies automatically. The mobile user can select a menu item by sending in another character. In this guide, you can read about the necessary configuration to start your own SMS information menu.
Download: SMS_information_menu.zip (2 KB)

Introduction
This solution is a simple, fast and automatized way to share information with customers. The character sent by the customer generates a constant reply which contains the needed information. All you have to do is to provide the characters or words the customer can select from and to configure your system.
How to configure your system
To launch SMS information menu via SMS text messages you need to build your own SMS system. To do so download and install Ozeki NG SMS Gateway to your computer. This software product will ensure SMS functionality and operate your SMS system. After the installation you can start to receive SMS messages from mobiles to your PC and send response messages from PC to mobiles.
Description of the solution
If a customer sends a message containing the chosen menu number to a predefined phone number, he will receive a welcome response message containing the description of the SMS information menu. In the response message two letters "i" and "e" are mentioned from which the customer can choose. Further information and options belong to each letter, from which the customer send back the one, he wants to get response message of. The guide below provides a detailed description of the system.
System architecture
After you built your SMS system it will work as follows: A customer sends an empty message to you. Ozeki NG SMS Gateway processes this message and its ASP user will check the phone number and its state in the database. The response to the message is determined by the state of the mobile phone. Figure 1 demonstrates this process.

Figure 1 - SMS Information Menu
1. Configuration
Step 1, MySQL
First you need to download and extracted sms_information_menu.zip file to your computer.
Open MySQL Command Line Client, log in with your password and create a table using the below script for sms information. Select the contents of the below and copy it into the MySQL Command Line Client. (Figure 2)

Create table script for MySQL

```
CREATE TABLE `numbers` (
  `id` int(11) NOT NULL auto_increment,
  `phonenum` varchar(50) NOT NULL,
  `state` varchar(50) NOT NULL,
  PRIMARY KEY (`id`)
)  DEFAULT CHARSET=utf8;
```

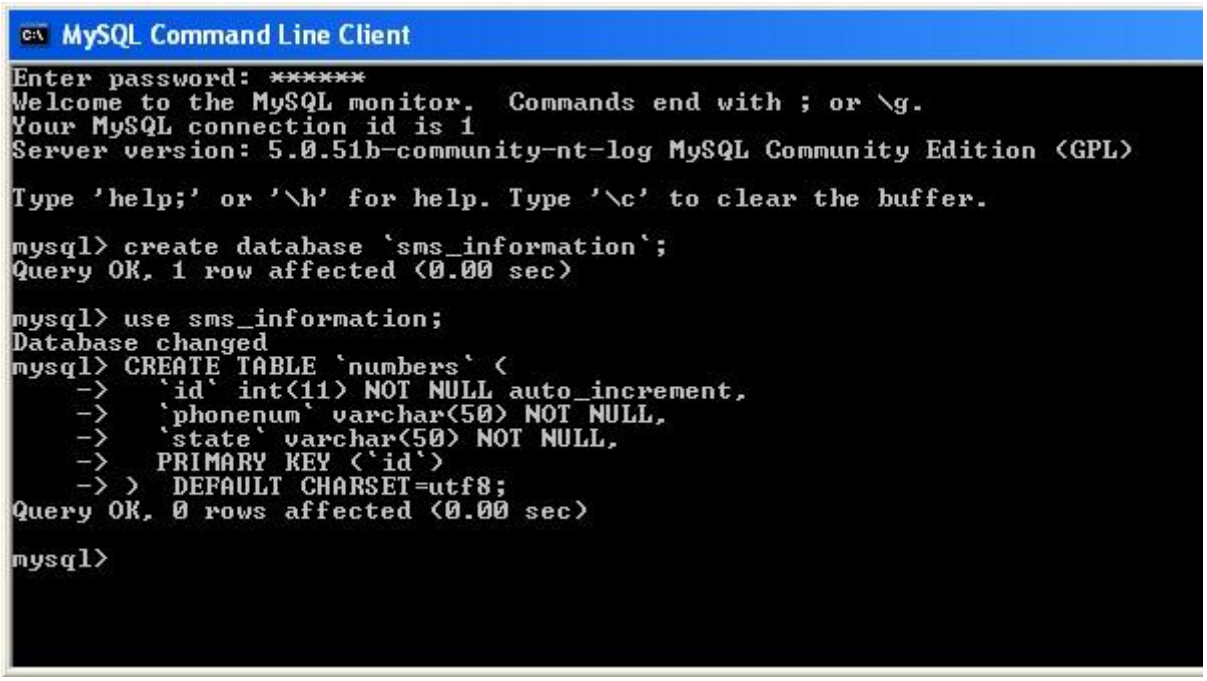


Figure 2 - MySQL Command Line

Step 2, Setup Ozeki NG
The downloaded "sms_information_menu.zip" file contains an ASPX file named "sms_information_menu.aspx". Open this file and search the line containing the connection string and customize it. Below you can see an example of customized connection string.
Connection string

```
Driver={MySQL ODBC 5.1 Driver};Server=127.0.0.1;

Database=MyInformation;User=root;Password=qwe123;Option=4;
```

Below you can see the source code of the "sms_information_menu.aspx" file.

```
<%@ Page Language="C#"%>
<%@ Import Namespace="System" %>
<%@ Import Namespace="System.IO" %>
<%@ Import Namespace="System.Collections" %>
<%@ Import Namespace="System.Collections.Generic" %>
<%@ Import Namespace="System.Data.SqlClient" %>
<%@ Import Namespace="System.Data.Odbc" %>

<%

//************************************************************
//copy the properties of the incoming SMS into local variables
//************************************************************

string sender = Request.QueryString["sender"];
string receiver = Request.QueryString["receiver"];
string messagedata = Request.QueryString["messagedata"];
string messageid = Request.QueryString["messageid"];
string messagetype = Request.QueryString["messagetype"];
string senttime = Request.QueryString["senttime"];
string receivedtime = Request.QueryString["receivedtime"];
string serviceprovider = Request.QueryString["operatornames"];

string respmsg = "";

//************************************************************
```

```
//Create the responses for the messages
//************************************************************

    //response message for the empty messagedata
        string RespForEmpty = "Welcome to the SMS information menu. If you want to receive
information about the import possibilities, send the character 'i', if you want to receive
information about export possibilities, send character 'e'";

        //response messages for letter 'i'
        Dictionary<string, string> RespForI =
              new Dictionary<string, string>();
        RespForI.Add("i", "1.public import \0x0D\0x0A2.cargo \0x0D\0x0A3.manifest \0x0D\0x0A
4.temp import \0x0D\0x0A5.warehouse \0x0D\0x0A6.re-import");

        RespForI.Add("i1", "1.public import1 \0x0D\0x0A2.public import2 \0x0D\0x0A
3.public import3 \0x0D\0x0A4.public import4 \0x0D\0x0A5.public import5                    \0x0D\0x0A
6.public import6 \0x0D\0x0A");
        RespForI.Add("i2", "2.cargo1 \0x0D\0x0A3.cargo2 \0x0D\0x0A4.cargo3 \0x0D\0x0A
4.cargo4 \0x0D\0x0A5.cargo5 \0x0D\0x0A6.cargo6 \0x0D\0x0A");
        RespForI.Add("i3", "1.manifest1 \0x0D\0x0A2.manifest2 \0x0D\0x0A
3.manifest3 \0x0D\0x0A4.manifest4 \0x0D\0x0A5.manifest5 \0x0D\0x0A6.manifest6 \0x0D\0x0A");
        RespForI.Add("i4", "1.temp import1 \0x0D\0x0A2.temp import2 \0x0D\0x0A
3.temp import3 \0x0D\0x0A4.temp import4 \0x0D\0x0A5.temp import5
\0x0D\0x0A6.temp import6 \0x0D\0x0A");
        RespForI.Add("i5", "1.warehouse1 \0x0D\0x0A2.warehouse2 \0x0D\0x0A
3.warehouse3 \0x0D\0x0A
4.warehouse4 \0x0D\0x0A5.warehouse5 \0x0D\0x0A6.warehouse6 \0x0D\0x0A");
        RespForI.Add("i6", "1.re-import1 \0x0D\0x0A2.re-import2 \0x0D\0x0A3.re-import3 \0x0D\0x0A
4.re-import4 \0x0D\0x0A5.re-import5 \0x0D\0x0A6.re-import6 \0x0D\0x0A");

        //response messages for letter 'e'
        Dictionary<string, string> RespForE =
              new Dictionary<string, string>();
        RespForE.Add("e", "1.public export. \0x0D\0x0A2.cargo \0x0D\0x0A3.re-
export \0x0D\0x0A4.temp re-export \0x0D\0x0A5.export manifest
\0x0D\0x0A6.countries");

        RespForE.Add("e1", "1.public export1 \0x0D\0x0A2.public export2 \0x0D
\0x0A3.public export3 \0x0D\0x0A4.public export4 \0x0D
\0x0A5.public export5 \0x0D\0x0A6.public
export6 \0x0D\0x0A");
        RespForE.Add("e2", "1.cargo1 \0x0D\0x0A2.cargo2 \0x0D\0x0A3.cargo3 \0x0D
\0x0A4.cargo4 \0x0D\0x0A5.cargo5 \0x0D\0x0A.6cargo6 \0x0D\0x0A");
        RespForE.Add("e3", "1.re-export1 \0x0D\0x0A2.re-export2 \0x0D\0x0A3.re-export3
\0x0D\0x0A4.re-export4 \0x0D\0x0A5.re-export5 \0x0D\0x0A6.re-export6 \0x0D\0x0A");
        RespForE.Add("e4", "1.temp re-export1 \0x0D\0x0A2.temp re-export2 \0x0D
\0x0A3.temp re-export3 \0x0D\0x0A4.temp re-export4 \0x0D
\0x0A5.temp re-export5 \0x0D\0x0A6.temp
re-export6 \0x0D\0x0A");
        RespForE.Add("e5", "1.export manifest1 \0x0D\0x0A2.export manifest2 \0x0D
\0x0A3.export manifest3 \0x0D\0x0A4.export manifest4 \0x0D\0x0A
5.export manifest5 \0x0D\0x0A6.export manifest6 \0x0D\0x0A");
        RespForE.Add("e6", "1.countries1 \0x0D\0x0A2.countries2 \0x0D\0x0A3.countries3
\0x0D\0x0A4.countries4 \0x0D\0x0A5.countries5 \0x0D\0x0A6.countries6 \0x0D\0x0A");


//************************************************************
//Respond to the messages
//************************************************************
        if (messagedata.Equals(""))
```

```
        {
            Response.Write("{SMS:TEXT:FORMATTED}{}{}{" + sender + "}
{" + RespForEmpty + "}");

        }
        else
        {
            string phonestate = messagedata.Trim().ToLowerInvariant();
            string getPhoneState = "select `state` from
`numbers` where `phonenum` = '" + sender + "'";

            try
            {
                string connectionString = "Driver={MySQL ODBC 5.1 Driver};Server=127.0.0.1;
Database=MyInformation;User=root;Password=qwe123;Option=4;";
                OdbcConnection oc = new OdbcConnection(connectionString);
                oc.Open();
                try
                {

                    try
                    {
                        string sqlresp = "";
                        OdbcCommand command = new OdbcCommand(getPhoneState, oc);
                        OdbcDataReader reader = command.ExecuteReader();
                        while (reader.Read())
                        {
                            sqlresp = reader.GetString(0);
                        }
                        reader.Close();

                        //if the phone number is not yet stored,
inserting it into the database
                        if (sqlresp == "")
                        {
                            string insertSQL =
"insert into `numbers` (`phonenum`, `state`) values                    ('" + sender + "','" + phonestate + "');";
                            OdbcCommand storephonenum = new OdbcCommand(insertSQL, oc);
                            storephonenum.ExecuteNonQuery();
                        }
                            //if the phone number is stored
in the datbase, modifying its phone state to                                  according the current messagedata
                            //and updating the corresponding record in the database
                        else
                        {
                            if (!(phonestate.EndsWith("i")
 || phonestate.EndsWith("e")))
                            {
                                phonestate = sqlresp + phonestate;
                                string updateSQL =
"update `numbers` set `state`= '"                    + phonestate + "' where `phonenum` = '" + sender + "';";
                                OdbcCommand updatephonestate = new OdbcCommand(updateSQL, oc);
                                updatephonestate.ExecuteNonQuery();
                            }
                            else
                            {
                                string updateSQL =
"update `numbers` set `state`= '"                    + phonestate + "' where `phonenum` = '" + sender + "';";
                                OdbcCommand updatephonestate = new OdbcCommand(updateSQL, oc);
                                updatephonestate.ExecuteNonQuery();
```

```
                    }

                }

                oc.Close();
            }
            catch (Exception z)
            {
                string errCode = z.Message;
                Response.Write(errCode);
            }

        }
        catch (Exception z)
        {
            string errCode = z.Message;
            Response.Write(errCode);
        }
    }
    catch (Exception z)
    {
        string errCode = z.Message;
        Response.Write(errCode);
    }

    //setting up the response messages for the recipient
    if (phonestate.StartsWith("i"))
    {
        respmsg = RespForI[phonestate];
    }

    if (phonestate.StartsWith("e"))
    {
        respmsg = RespForE[phonestate];
    }
    string respsms = "{SMS:TEXT:FORMATTED}{}{}{" + sender + "}
{" + respmsg + "}";
    Response.Write(respsms);
  }

%>
```

You can customize the responses by modifying the following variables:
RespForEmpty: this variable contains the response message for an empty message. In this example it is a description for the menu.
RespForI: this variable contains the response message for a message containing the letter "i".
RespForE: this variable contains the response message for a message containing the letter "e".
The "RespForI" and "RespForE" variables determines response messages for the chosen sub-menus (for example: "e4"). You can customize these response messages as well or you can add further sub-menus (for example: "e41").

After you customized the ASPX file, start Ozeki NG SMS Gateway and login with your username and password and create an ASP user. To do so click on "Add users and applications" and select "ASP" user in the list and click on "Install". Give the access path of sms_information_menu.aspx file in the ASP user's configuration form as in Figure 3.

Figure 3 - ASP script file

After configuration, your system is ready to use the SMS Information Menu.
2. Testing the system
First time I simulate an incoming message which message data is empty. You can see the response message to the empty message in Figure 4, as described in the "RespForEmpty" variable.

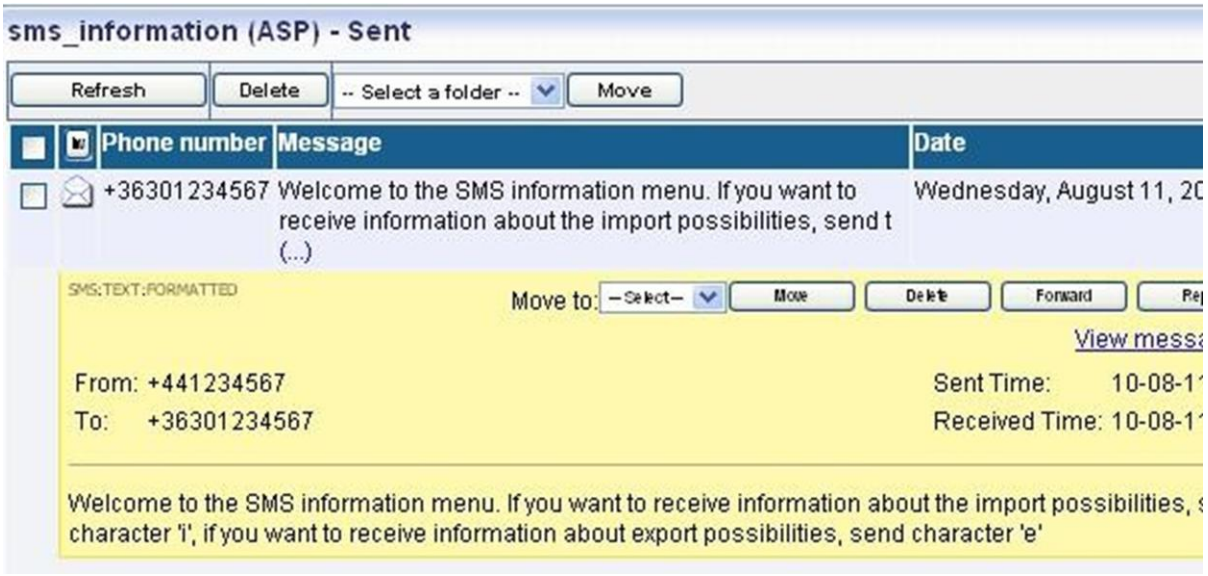


Figure 4 - Response to empty SMS

In Figure 5 you can see the response message for letter "i", as described in the "RespForI" variable's "i" keyword:

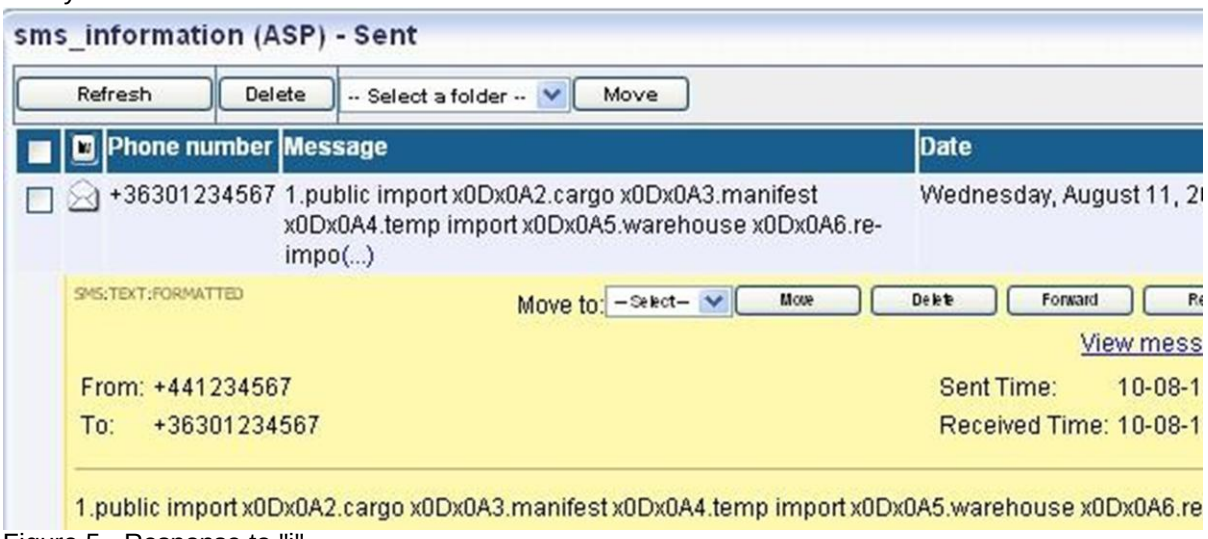


Figure 5 - Response to "i"

If I send the number "3" I will get the response for number "3" at the "i" menu, as described in the "RespForI" variable's "i3" keyword:

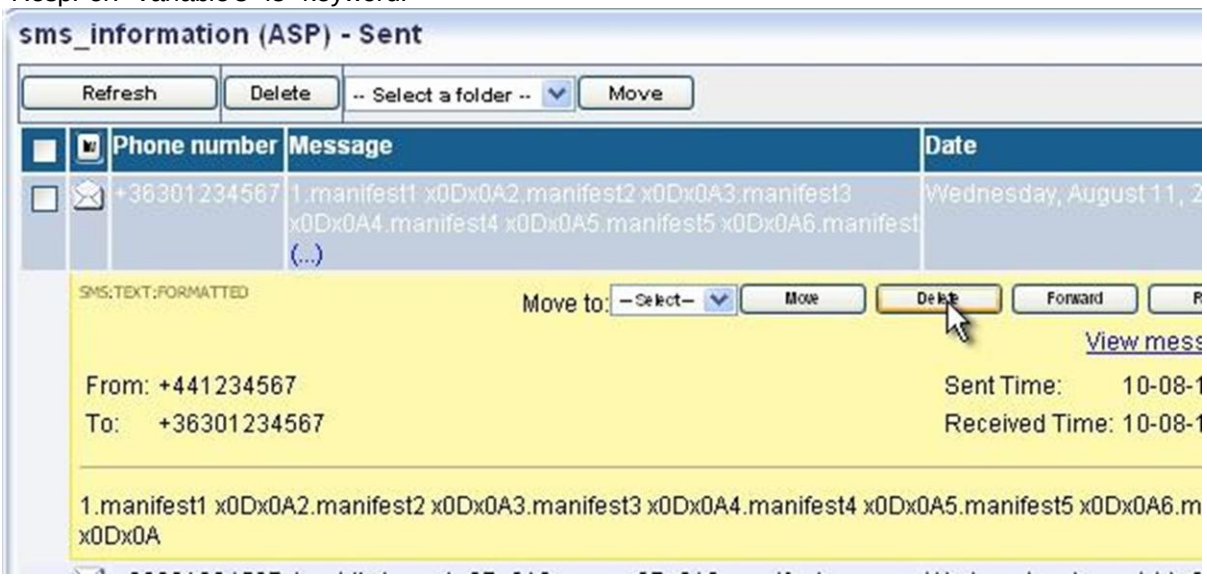


Figure 6 - Response to "i3"

In Figure 7 you can see the response message for letter "e", as described in the "RespForE" variable's "e" keyword:

Figure 7 - Response to "e"

If I send the number "4" I will get the response for number "4" at the "e" menu, as described in the "RespForE" variable's "e4" keyword:

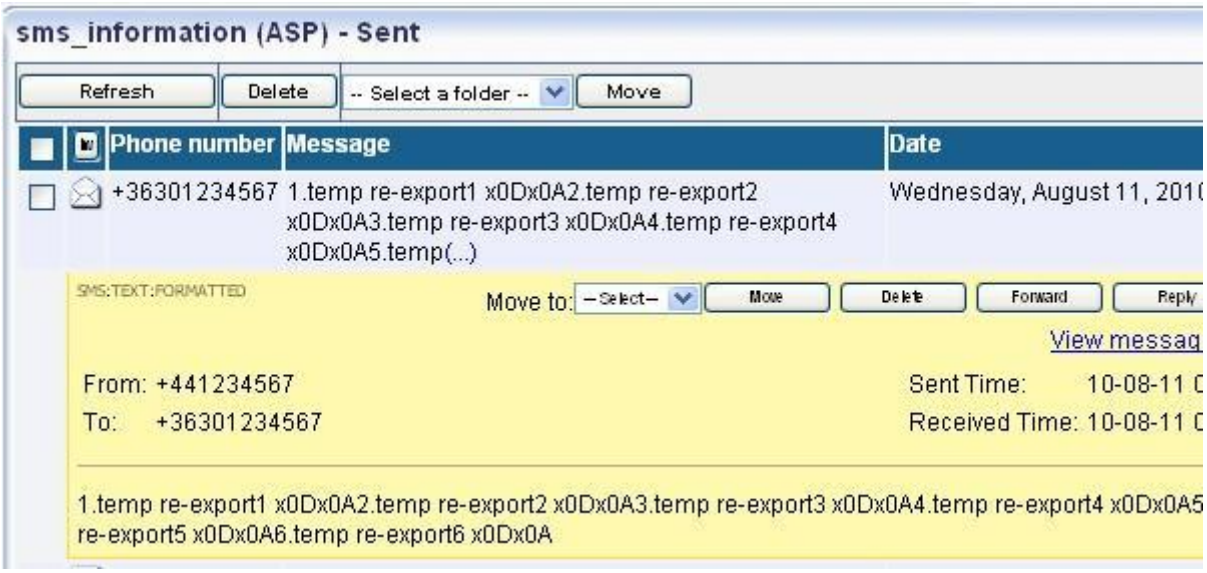


Figure 8 -Response to "e4"

7. How to introduce SMS reminders in your website with QuesCom GSM Gateway and Ozeki NG SMS Gateway

When you use Ozeki NG SMS Gateway software you can introduce various SMS services and solutions. One of these solutions is adding SMS reminder form to your website. This SMS reminder form enables your website visitors to subscribe to your various SMS reminders while browsing your website. Follow the configuration steps below to add the SMS reminder form to your website in minutes!

Download: ozeki_reminder_example.zip

With Ozeki NG SMS Gateway you can easily introduce an SMS reminder form in your website. By following just a few quick steps below you can make the SMS reminder form available for your website visitors.

Then people can subscribe to your SMS reminders while browsing your website. You can specify various events for which people can ask an SMS reminder. For example, they can ask a reminder via SMS before their payment is in due. This solution is convenient and discreet for both sides!

How it works?

For getting SMS reminders, website visitors only need to fill-in the form on your website. They only need to specify their phone number, the exact time and type of the SMS reminder. Finally, just click Create Reminder.

This SMS reminder then will be inserted in the ozekimessageout database table of Ozeki NG SMS Gateway. By default, Ozeki NG SMS Gateway checks ozekimessageout table in every 10 sec. for outgoing messages. When the given SMS reminder is polled, Ozeki NG SMS Gateway sends out the SMS reminder to the recipient.

Getting started

For getting started first you need to configure Ozeki NG SMS Gateway for sending SMS messages (if you haven't done it): Follow the Quick Start Guide.

Setup QuesCom GSM gateway and Ozeki link (new page to be created, content at the end of that document)

Then you need to configure a database user in Ozeki NG SMS Gateway. In this example, MySQL database is used. For configuration steps on how to create MySQL database user, check MySQL overview page.

When your SMS system is configured, download ozeki_reminder_example.zip and follow the configuration steps below.

Configuration steps

First download and extract ozeki_reminder_example.zip onto your PC (Figure 1).

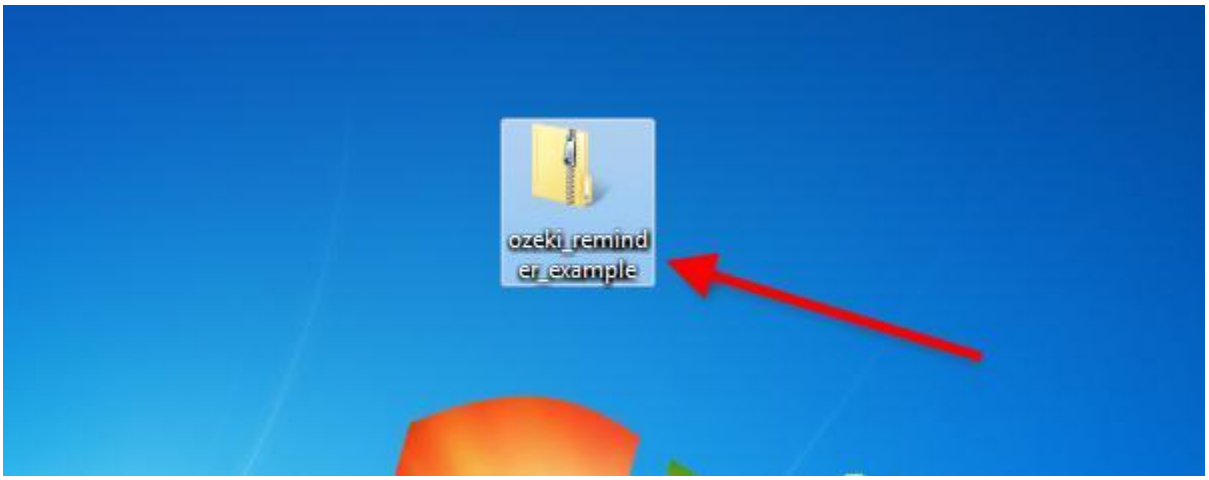


Figure 1 - Download Ozeki SMS reminder

Open Ozeki Reminder Example file and copy the Ozeki folder in it (Figure 2).

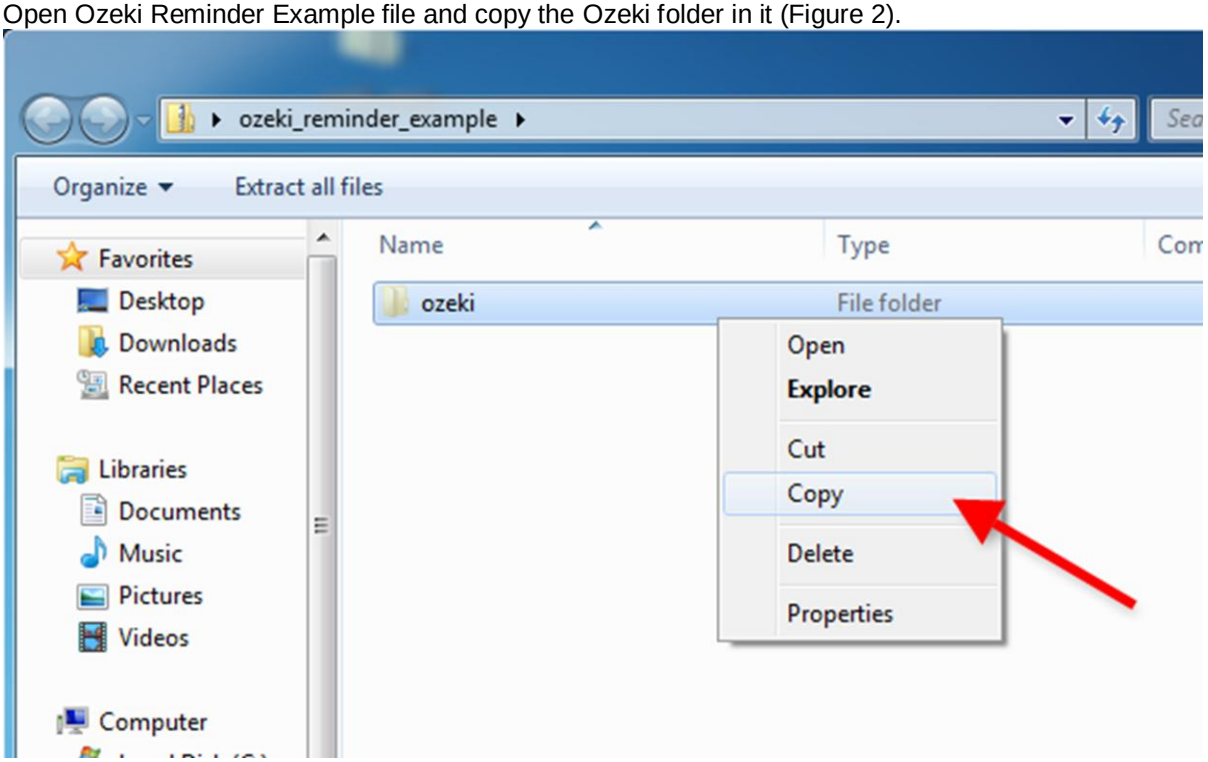


Figure 2 - Copy Ozeki folder

Paste the Ozeki folder into the folder of your webserver (Figure 3).

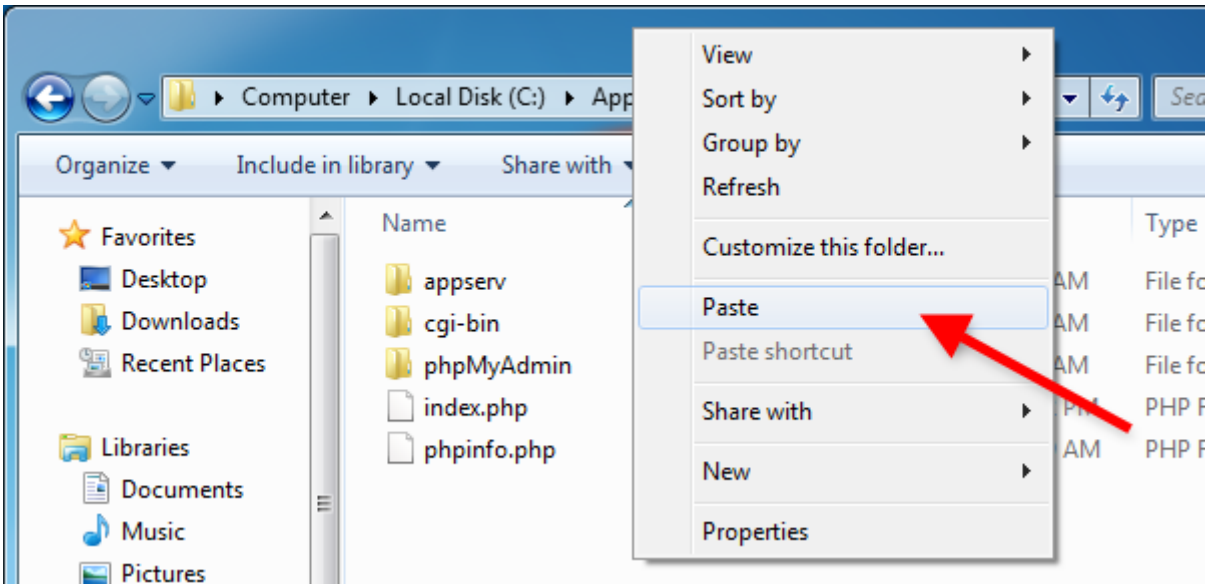


Figure 3 - Paste Ozeki folder into the webserver folder

Open ozekiDB.class.php in Ozeki folder (Figure 4). Customize the following variables according to your database:

$dbHost
$dbName
$dbUserName
$dbPassword

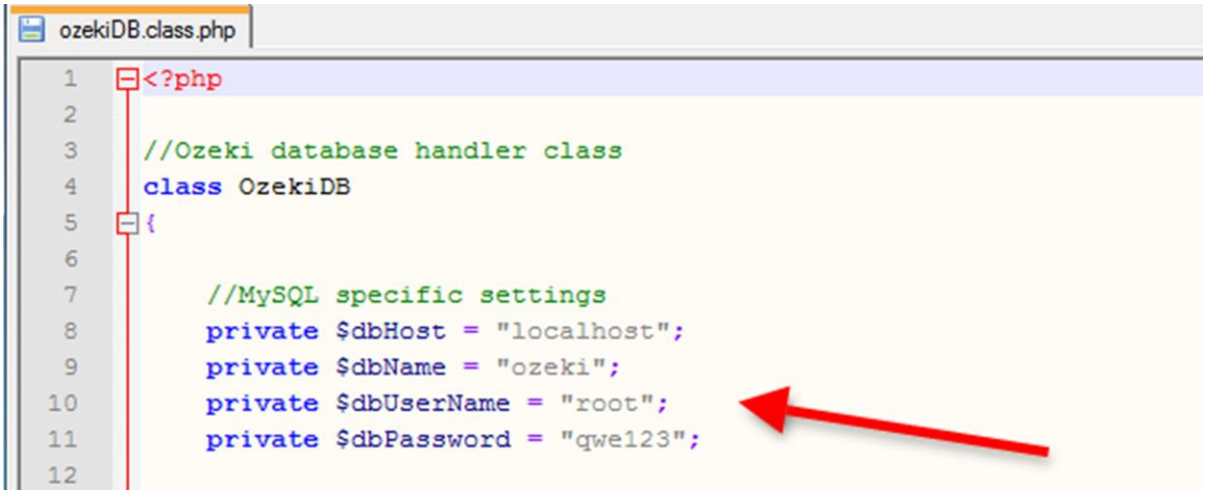


Figure 4 - Customize database variables

Now a 'sendondate' database column needs to be added to ozekimessageout table. For this purpose execute the following statement (Figure 5):

```
ALTER TABLE `ozekimessageout` ADD `sendondate` VARCHAR( 20 ) NOT NULL ;
```

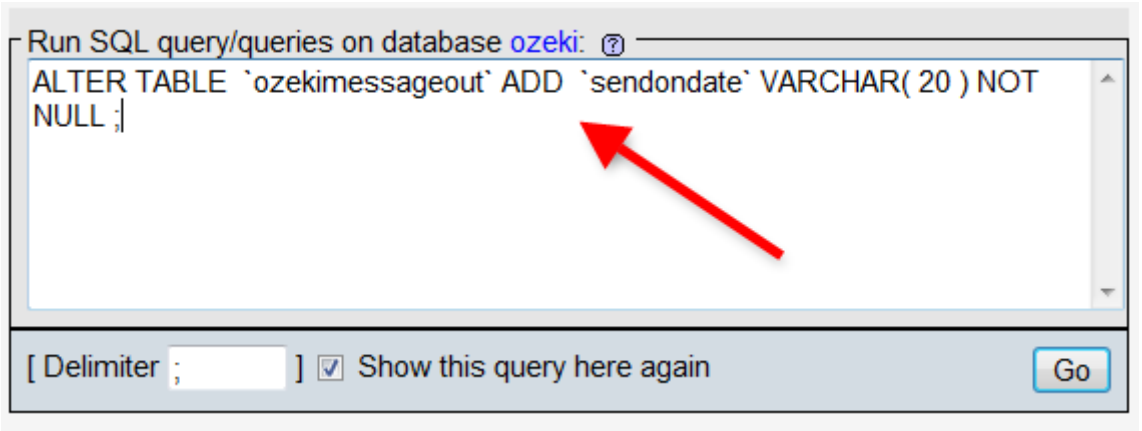


Figure 5 - Add sendondate column

Start Ozeki NG SMS Gateway and go to the configuration panel of the database user. Click on SQL for sending tab (Figure 6). In Polling tab insert the following SQL statement for polling messages:

```
SELECT `id`,`sender`,`receiver`,`msg`,`msgtype`,`operator` from `ozekimessageout` where ((EXTRACT(YEAR FROM (`sendondate`))) = YEAR( NOW( ))) AND ((EXTRACT(MONTH FROM (`sendondate`))) = MONTH( NOW( ))) AND ((EXTRACT(DAY FROM (`sendondate`))) = DAY( NOW( ))) AND ((EXTRACT(HOUR FROM (`sendondate`))) = HOUR( NOW( ))) AND ((EXTRACT(MINUTE FROM (`sendondate`))) = MINUTE( NOW( )))
```

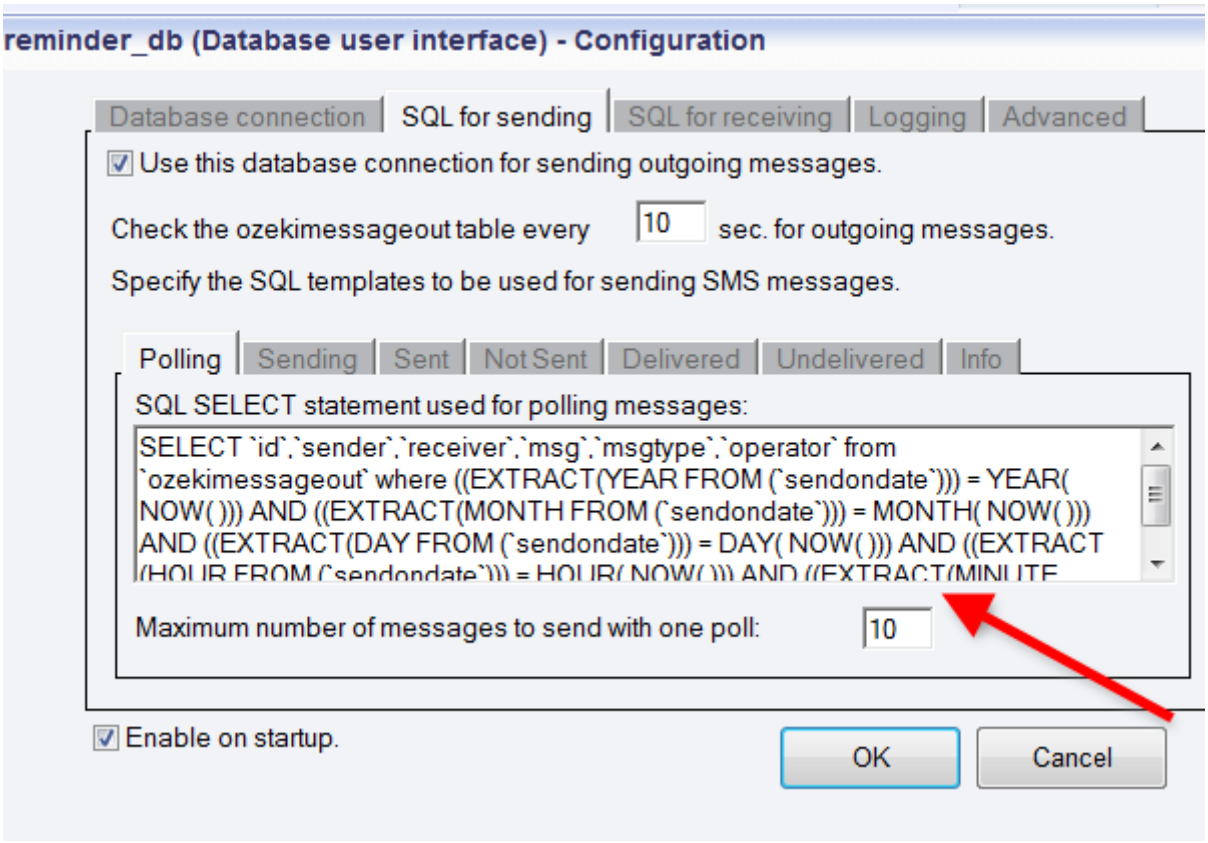


Figure 6 - SQL statement for polling messages

After these configuration steps you can open your website to which you have added the SMS reminder form. For testing, fill-in the fields and click on Create Reminder (Figure 7).

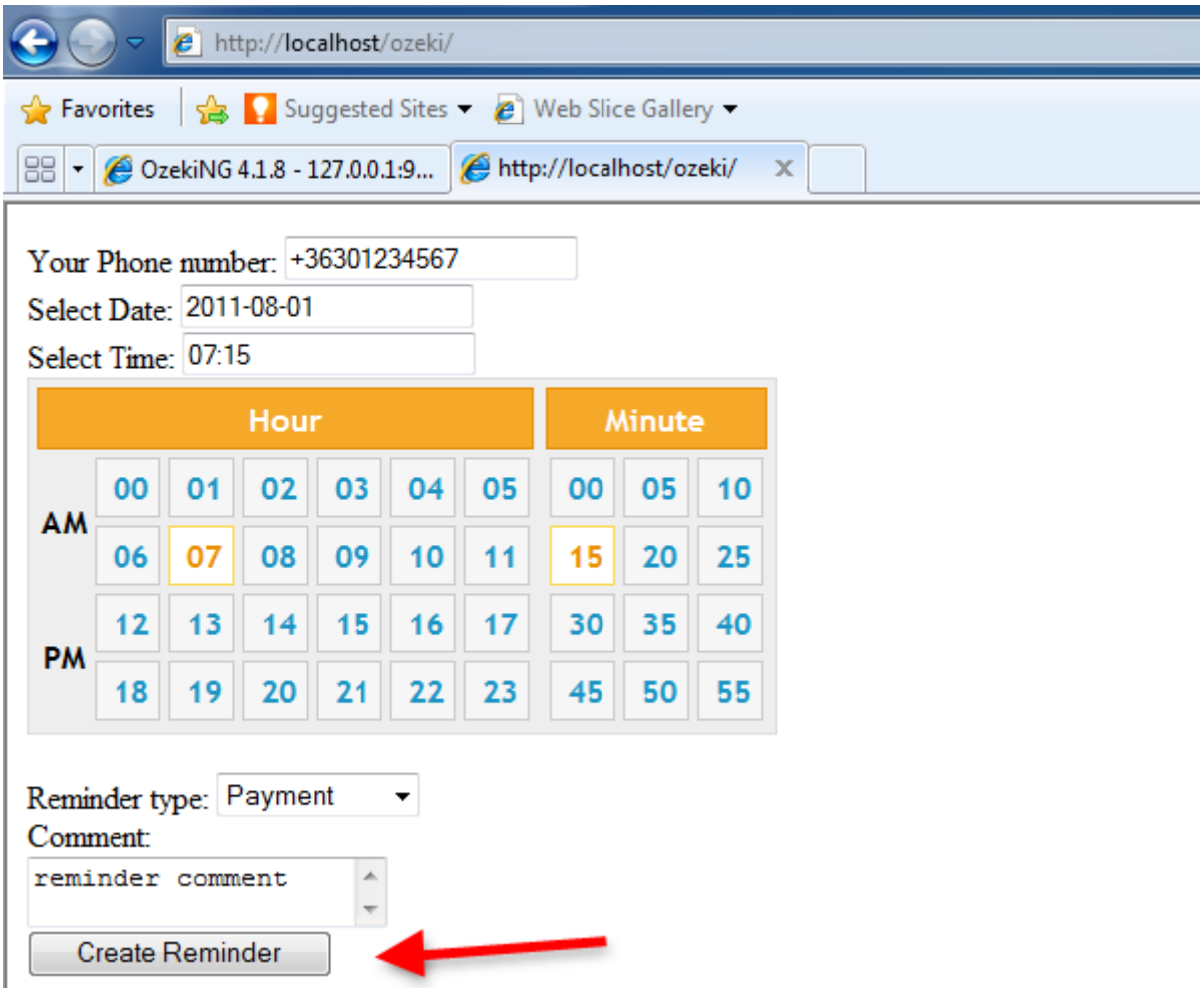


Figure 7 - Test

Figure 8 demonstrates that Ozeki NG SMS Gateway have found 1 outgoing message and delivers it on the requested date and time.

```
8/1/2011 7:14:46 AM - INFO 3535: Outgoing message found in database (total 0 messages
8/1/2011 7:14:56 AM - INFO 3535: Outgoing message found in database (total 0 messages
8/1/2011 7:15:06 AM - INFO 3535: Outgoing message found in database (total 1 messages
8/1/2011 7:15:06 AM - INFO 3511: Message accepted for delivery from user 'reminder_db
8/1/2011 7:15:06 AM - INFO 3512: Message successfully sent through connection: HTTPSe
8/1/2011 7:15:06 AM - INFO 3532: Message delivered to network. Updating correspondig
```

Figure 8 - Outgoing message is found and delivered

On Figure 9 you can see the sent SMS reminder message in the Sent folder of Ozeki NG SMS Gateway.

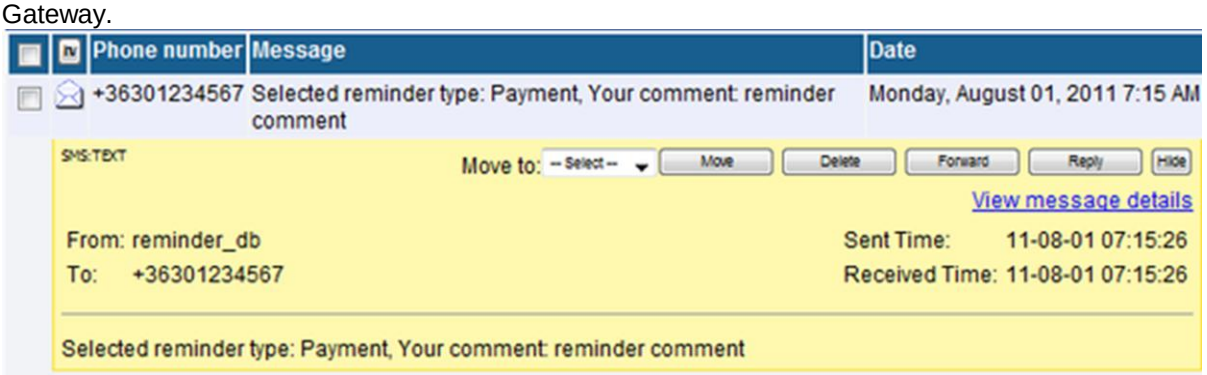


Figure 9 - Sent message

8. How to setup Ozeki Bulk SMS Client for sending bulk SMS with QuesCom GSM Gateway

This quick guide explains how you can setup Ozeki Bulk SMS Client in order to send bulk SMS messages with Ozeki NG SMS Gateway effectively. Ozeki Bulk SMS Client and an example CSV file can be downloaded below this site.

Download: ozeki_bulk_sms_client.zip
example_csv_file.csv

Ozeki Bulk SMS Client for Ozeki NG SMS Gateway is a powerful application. With this client, you will be able to send one text message to many recipients. The number of recipient phone numbers is unlimited, which means that you can send your SMS message to thousands or millions of people.

Ozeki Bulk SMS Client can be used to send SMS messages in large quantity simultaneously. This outstanding solution is recommended for companies and organizations who require to send bulk SMS in each month efficiently.

Recipient phone numbers can be listed in a CSV file that can be uploaded into the client. Then you can compose the body of the message and just click on Send. The client then forwards the bulk SMSs to Ozeki NG SMS Gateway via HTTP.

Configuration steps

It is assumed that you have already downloaded and configured Ozeki NG SMS Gateway for SMS messaging.

Setup QuesCom GSM gateway and Ozeki link (new page to be created, content at the end of that document)

Now you only need to download ozeki_bulk_sms_client.zip and extract it. Then execute Ozeki Bulk SMS Client (Figure 1).

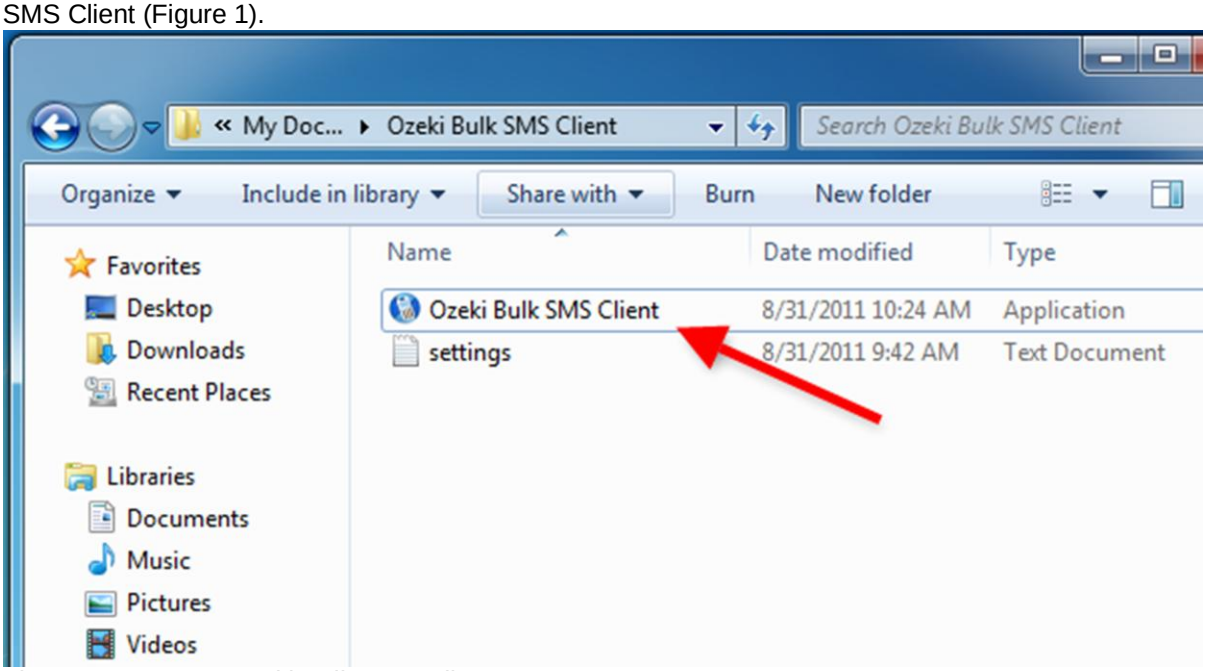


Figure 1 - Execute Ozeki Bulk SMS Client

Next click on Options to set Ozeki NG SMS Gateway settings (Figure 2).

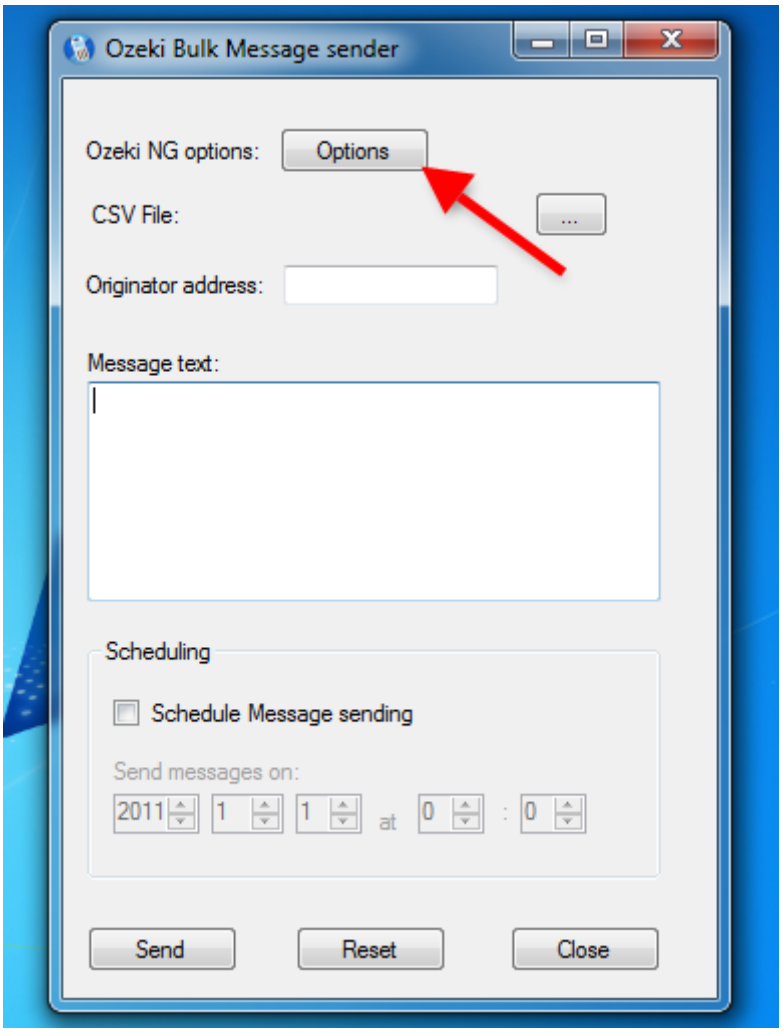


Figure 2 - Options

In Ozeki NG Options window you need to specify the following parameters:

IP address: Enter the IP address of the computer on which Ozeki NG SMS Gateway has been installed.
Port number: Enter the port number Ozeki NG SMS Gateway will listen to.
Username: Enter the name of the user via which you wish to send out bulk SMS messages in Ozeki NG SMS Gateway. In this example this is user admin.
Password: Enter the password you use for the selected user in Ozeki NG SMS Gateway.

Finally, click on Save (Figure 3).

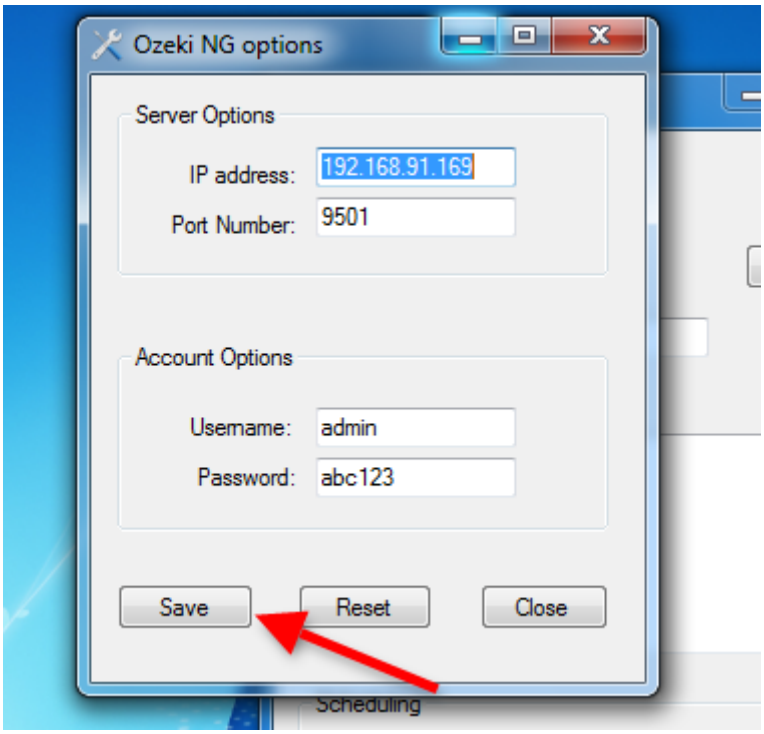


Figure 3 - Ozeki NG Options

Now you need to upload the CSV file containing the recipient phone numbers. To do so click on browse button next to CSV file (Figure 4).

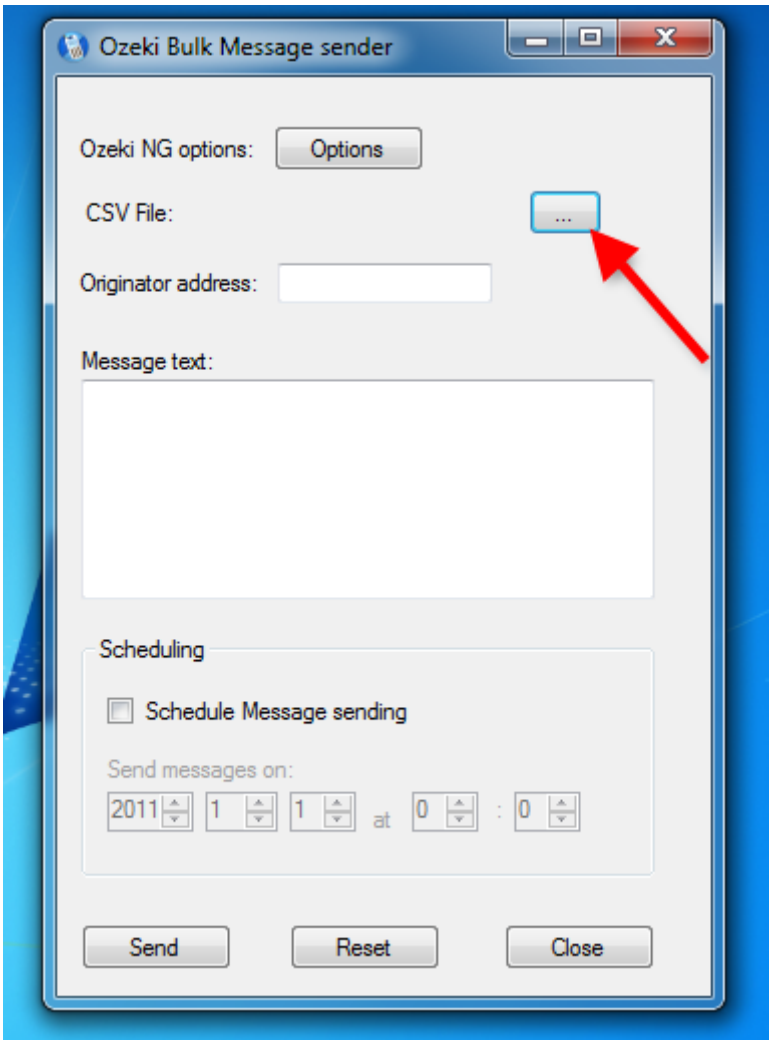


Figure 4 - Browse CSV file

Select the CSV file to be uploaded and click on Open (Figure 5). You can use the example_csv_file.csv for testing. The CSV must include only phone numbers separated by semicolon (;).

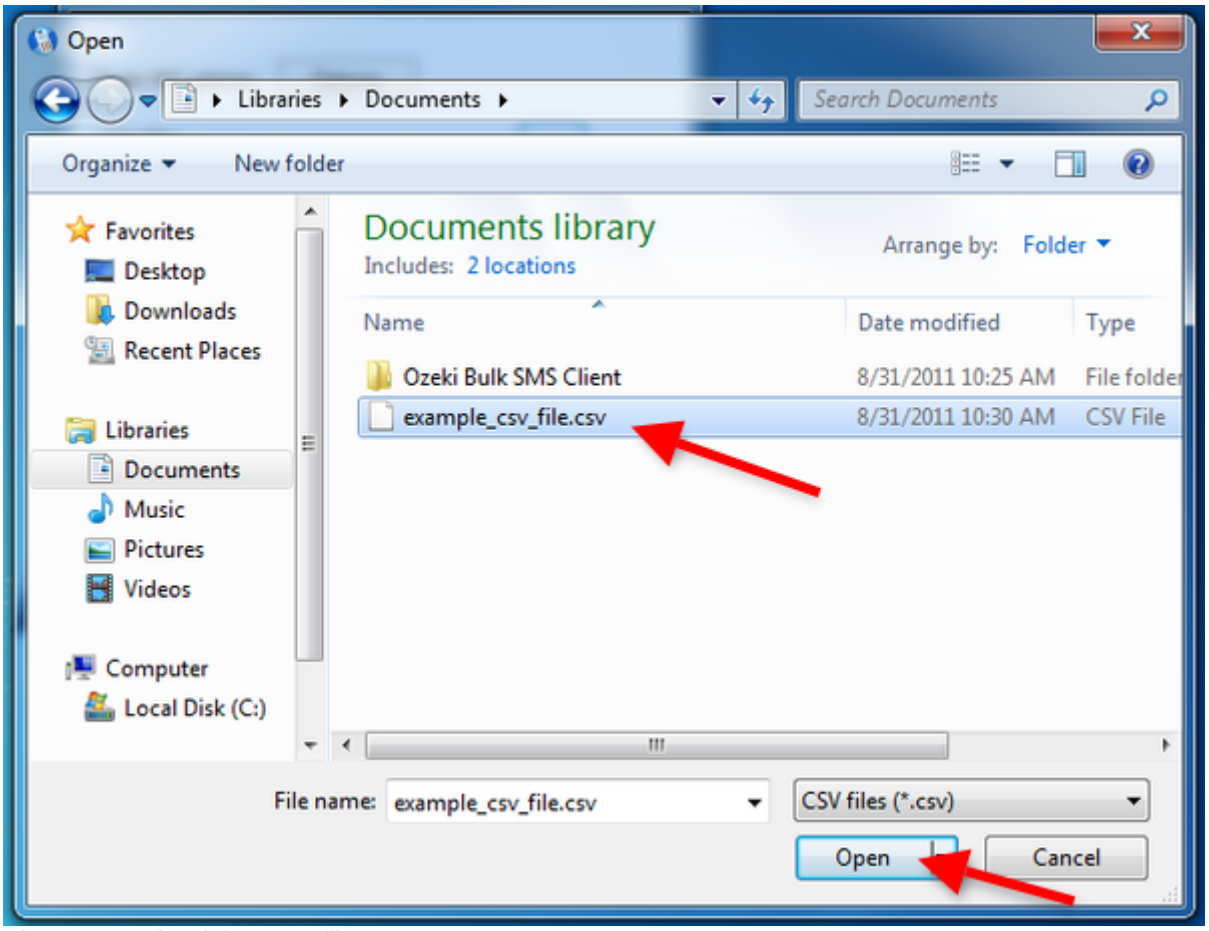


Figure 5 - Upload the CSV file

To send out bulk SMSs enter the Originator address. Originator address is the sender address that will be displayed on the receiver device in the sender field.

Then compose the body of the text message. If it is needed you can also schedule bulk SMS sending. For this purpose enable Schedule message sending checkbox and specify the Year - Month - Day - Hour - Minute when the bulk SMSs need to be sent.

Finally, just click on Send (Figure 6).

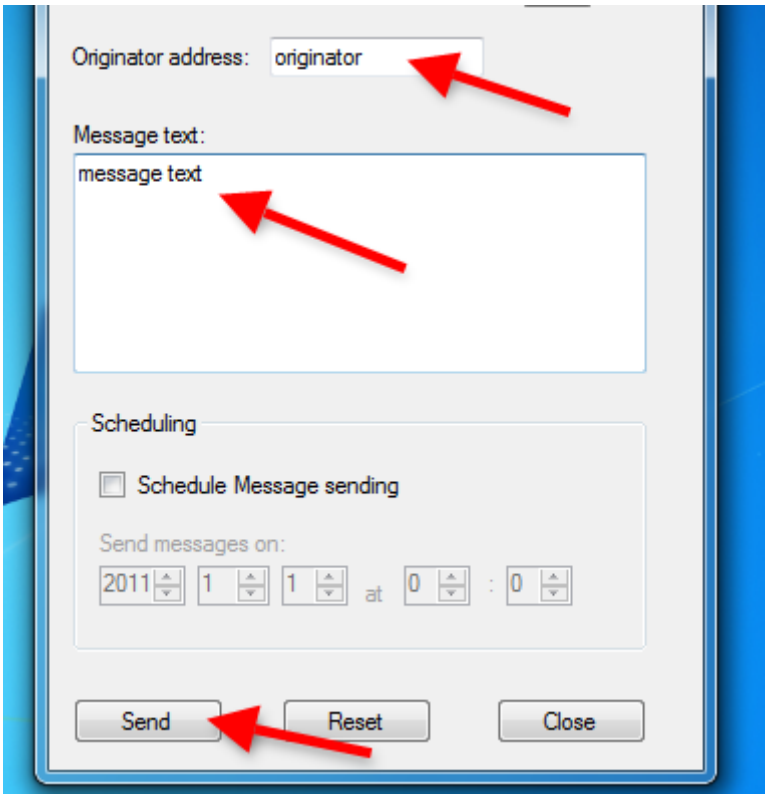


Figure 6 - Bulk SMS composing

Now the CSV file is loaded and bulk SMSs will be forwarded to Ozeki NG SMS Gateway via HTTP. Then Ozeki NG SMS Gateway sends out the messages to the listed recipients via the previously selected user. When messages are forwarded to Ozeki NG SMS Gateway successfully, a notification message will inform you about this fact (Figure 7).

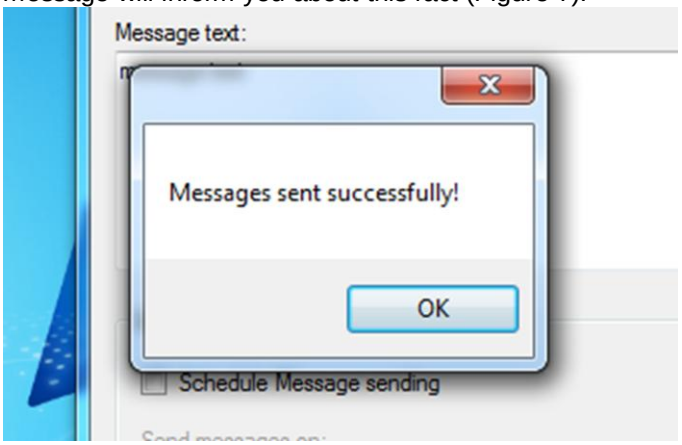


Figure 7 - Messages forwarded successfully

You can check the sent SMSs in the Sent folder of Ozeki NG SMS Gateway (Figure 8).

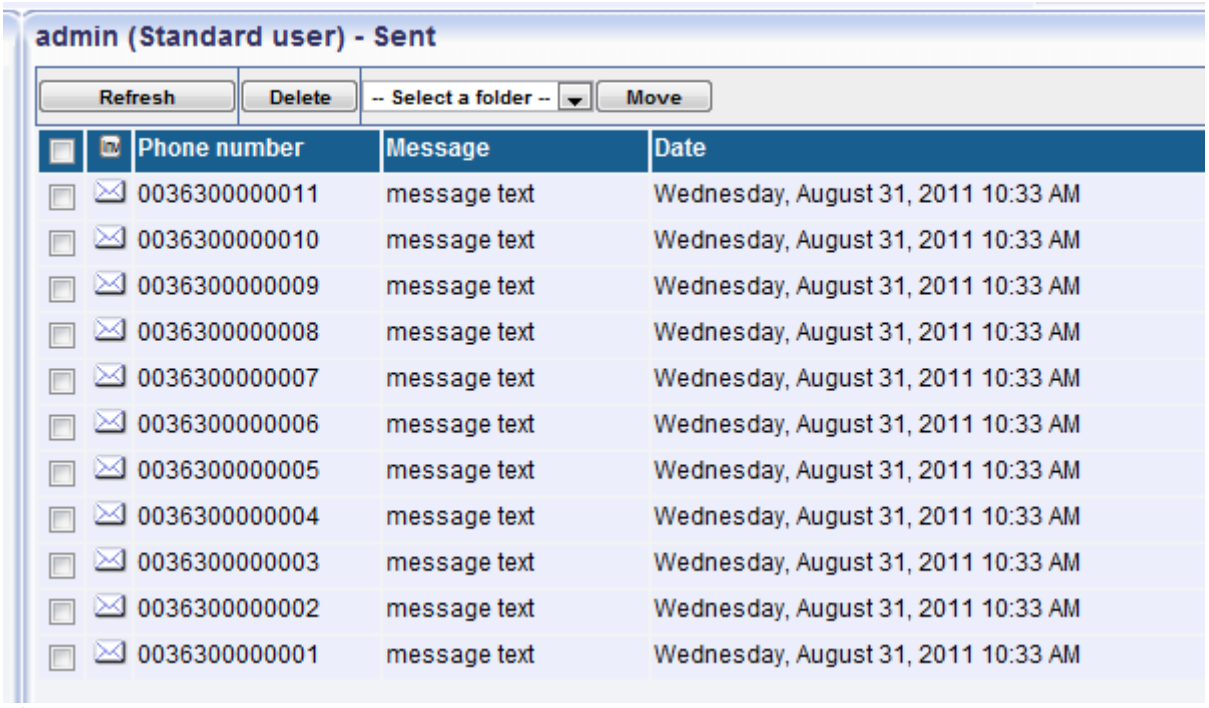


Figure 8 - Sent messages

9. Sport betting service via SMS

Find an SMS sport betting example on this page that will demonstrate how to build your own SMS system to introduce SMS betting service for your customers. By downloading the project (available below this page) you can start the betting service via SMS to make your business more attractive and let more people know about it. To implement this solution effectively and properly please follow the configuration guide and watch the video tutorial, as well.

Download: ozeki-sms-bet-example.zip

Introduction

SMS sport betting service practically means that users can bet to the outcome of a sport event using their mobile phones. The greatest advantage of this solution is its simplicity since there is no need for paper forms but people can send bets via SMS text messages. In this way you can collect their phone numbers even for further mobile marketing purposes or you can achieve that more people will know your business through your entertaining solution.

How this system works

To be able to implement SMS sport betting service, first, you need to download and install Ozeki NG SMS Gateway. This software will provide SMS functionality for your IT environment. The SMS gateway connects to the mobile network in two ways. It can operates one or more QuesCom GSM Gateway.

After you configure the SMS gateway SMS betting service will work as follows: Users need to send SMS text messages to a predefined phone number. Ozeki NG SMS Gateway will process these messages with the help of its ASP user (Learn how to create an ASP user in Ozeki NG SMS Gateway: ASP user setup). This ASP user forwards messages to a MySQL database to be stored. Finally, the results will appear on the web site (Figure 1). Settings can also be made on the web site.

Please note that the example below is made for MySQL database but other databases can also be used with this solution. Ozeki NG SMS Gateway connects to the database via ODBC driver.

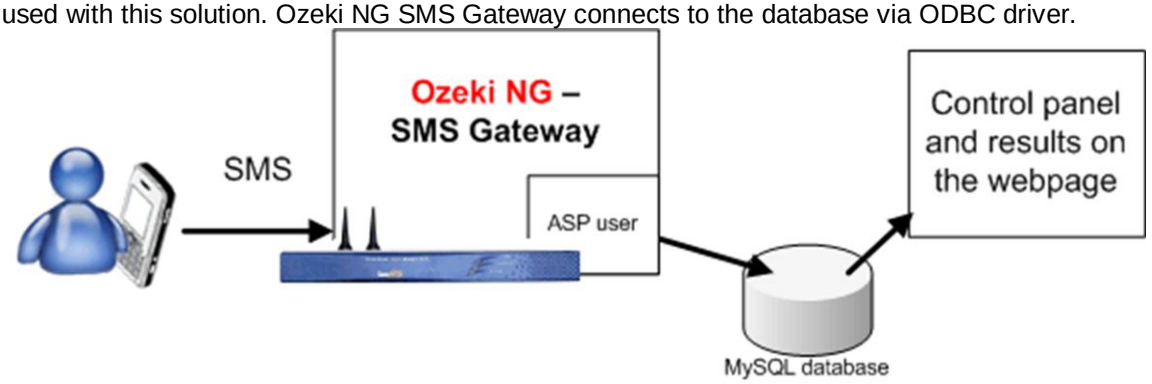


Figure 1 - System architecture

Configuration guide

I have already downloaded and extracted ozeki-sms-bet-example.zip file to the desktop. Figure 1 demonstrates the content of this file: webpage directory, MySQL-table-structure.txt file, and sms-bet.aspx file.

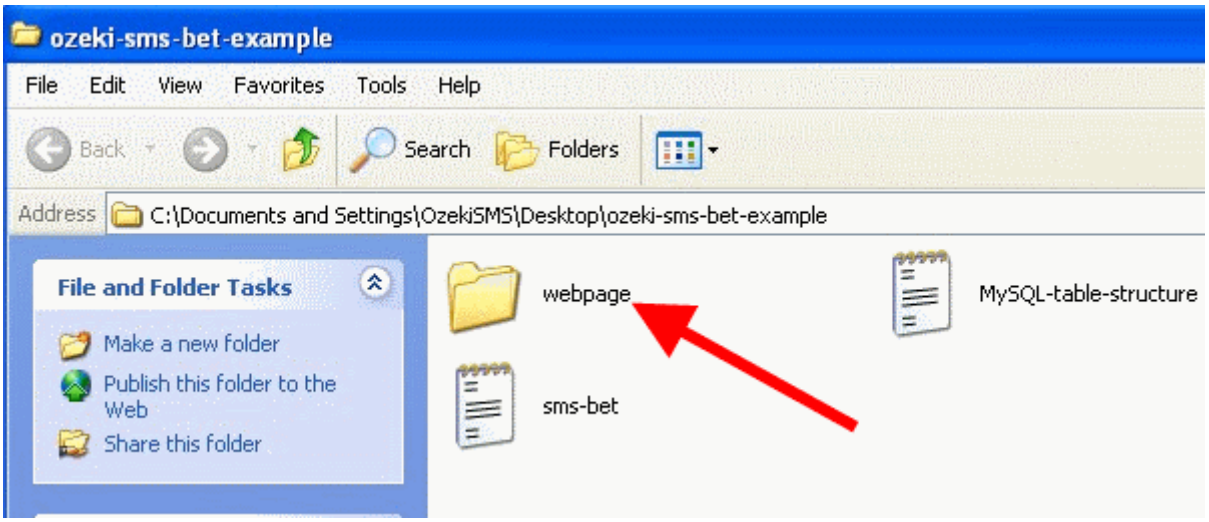


Figure 2 - Content of ozeki-sms-bet-example.zip file

Copy the content of webpage to the directory of the web page (wwwroot). I created an sms-bet directory and copied webpage into this directory so the full path in this example is: C:\AppServ\www\sms-bet (Figure 2).

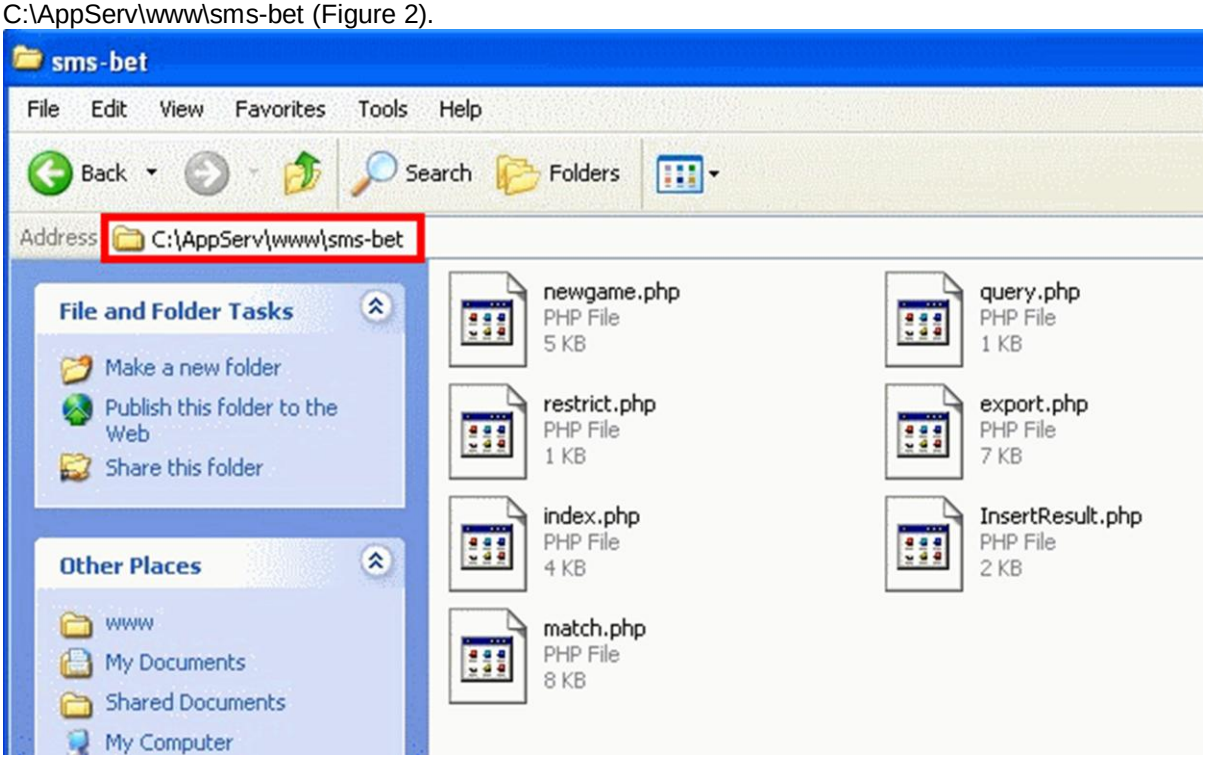


Figure 3 - Copy webpage directory

Open MySQL-table-structure.txt file (Figure 4).

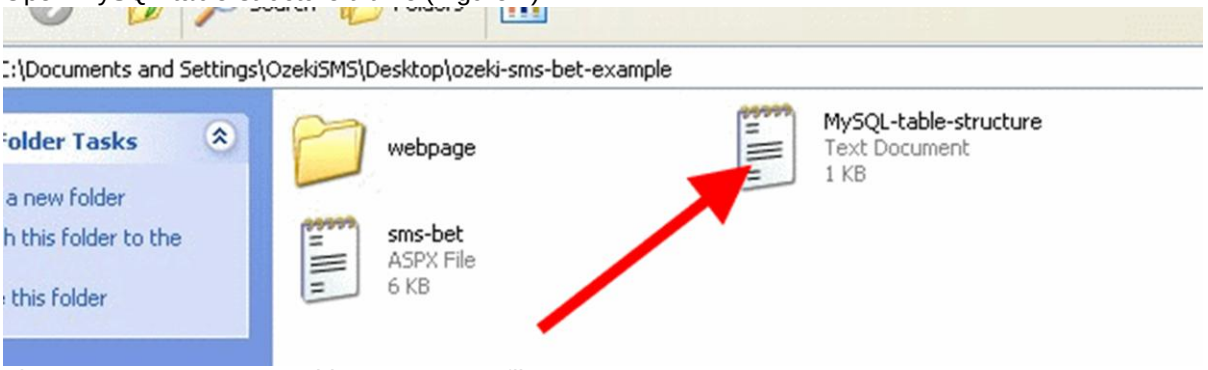


Figure 4 - Open MySQL-table-structure.txt file

MySQL-table-structure.txt file

```
CREATE DATABASE `bets`;

USE `bets`;

CREATE TABLE `bets` (
  `id` int(20) NOT NULL auto_increment,
  `sender` varchar(20) NOT NULL,
  `originalmsg` varchar(160) NOT NULL,
  `formatted_msg` varchar(160) NOT NULL,
  `match_number` int(5) NOT NULL,
  `team1` varchar(30) NOT NULL,
  `team2` varchar(20) NOT NULL,
  `time_of_bet` timestamp NOT NULL default CURRENT_TIMESTAMP,
  `msgid` varchar(30) NOT NULL,
  PRIMARY KEY  (`id`)
);


CREATE TABLE `matches` (
  `id` int(10) NOT NULL auto_increment,
  `match_number` varchar(10) NOT NULL,
  `match_begin` bigint(15) NOT NULL,
  `team1` varchar(30) NOT NULL,
  `team2` varchar(30) NOT NULL,
  `bet_begin` bigint(15) NOT NULL,
  `bet_end` bigint(15) NOT NULL,
  `final_result` varchar(40) NOT NULL,
  PRIMARY KEY  (`id`)
);

CREATE TABLE `settings` (
  `id` int(10) NOT NULL auto_increment,
  `name` varchar(10) NOT NULL,
  `value` tinyint(1) NOT NULL,
  PRIMARY KEY  (`id`)
);


INSERT INTO `settings` VALUES (1, 'restrict', 1);
```

Now copy the content of MySQL-table-structure.txt and paste it to the SQL console. The content of the file will be executed (Figure 5).

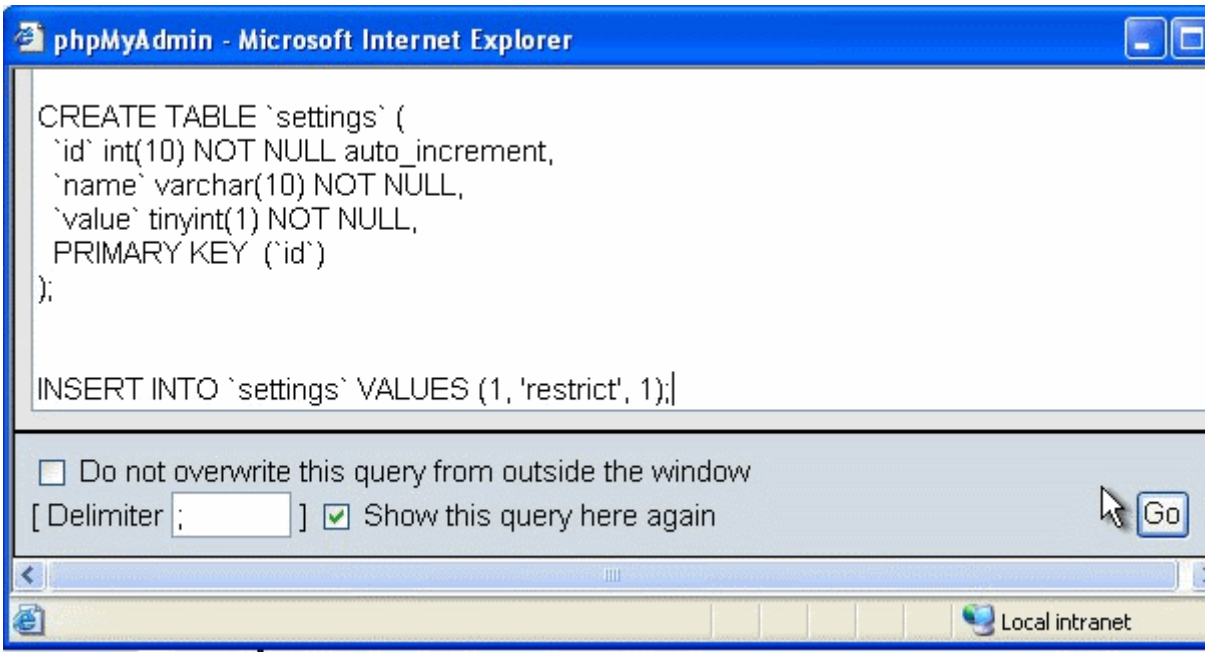


Figure 5 - Execute the content of MySQL-table-structure.txt

Copy the content of sms-bet.aspx file (Figure 6).

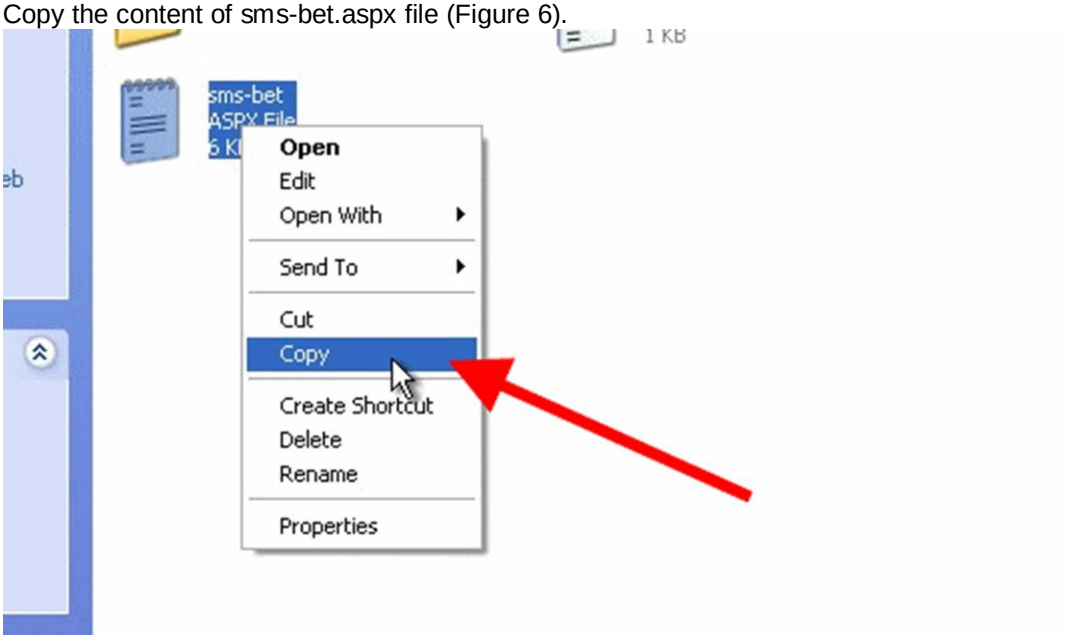


Figure 6 - Copy sms-bet.aspx

And paste sms-bet.aspx to its final place. In our example it is C:\bet\sms-bet.aspx (Figure 7).

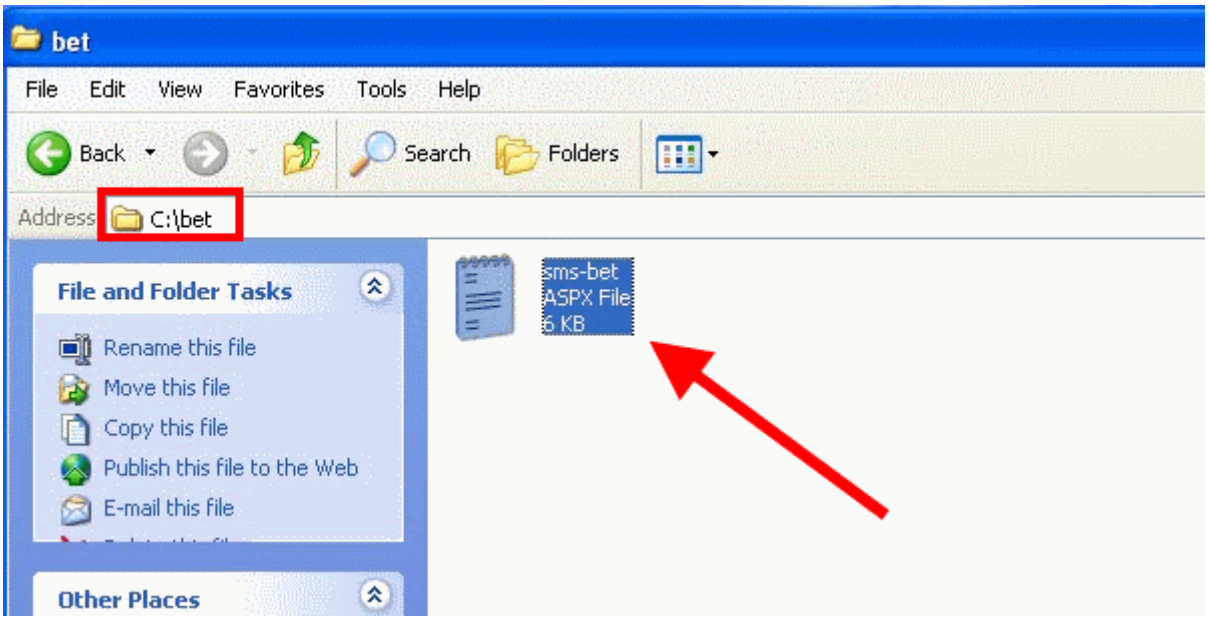


Figure 7 - Paste sms-bet.aspx

Install an ASP user in Ozeki NG SMS Gateway. On the Configuration pane of the user specify the path to the aspx file. In this example it is C:\bet\sms-bet.aspx (Figure 8).

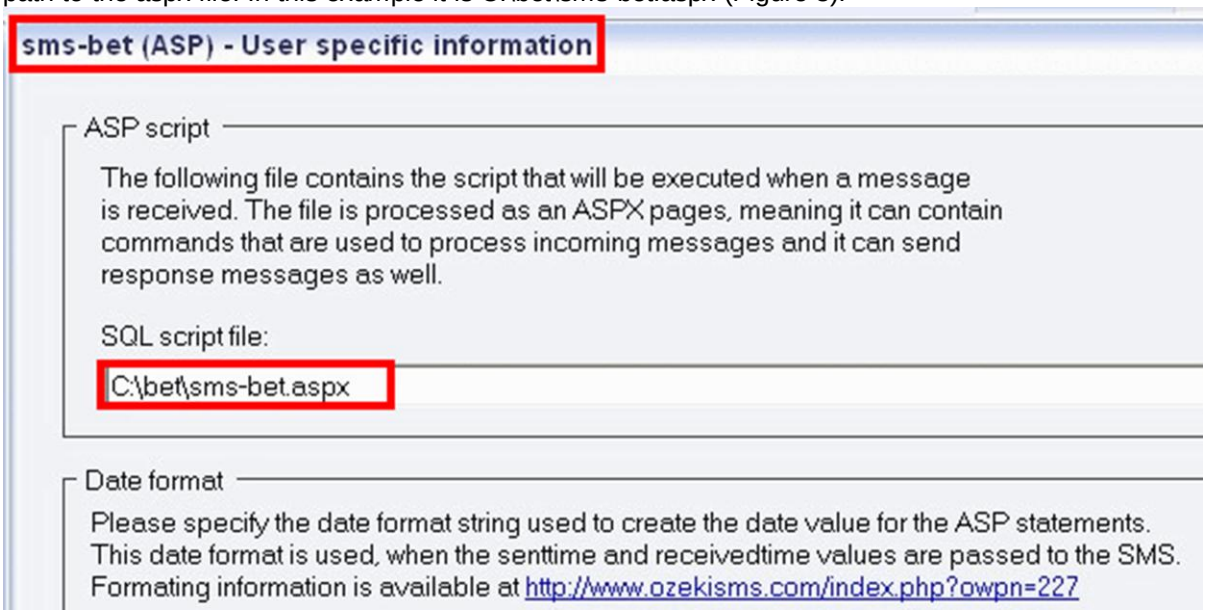


Figure 8 - Path to sms-bet.aspx file

Open sms-bet.aspx file to be able to customize it. You can specify the server, the database, the user, and its password (Figure 9). In our example these parameters are the follows:
Server: 127.0.0.1
Database: bets
User: root
Password: qwe123

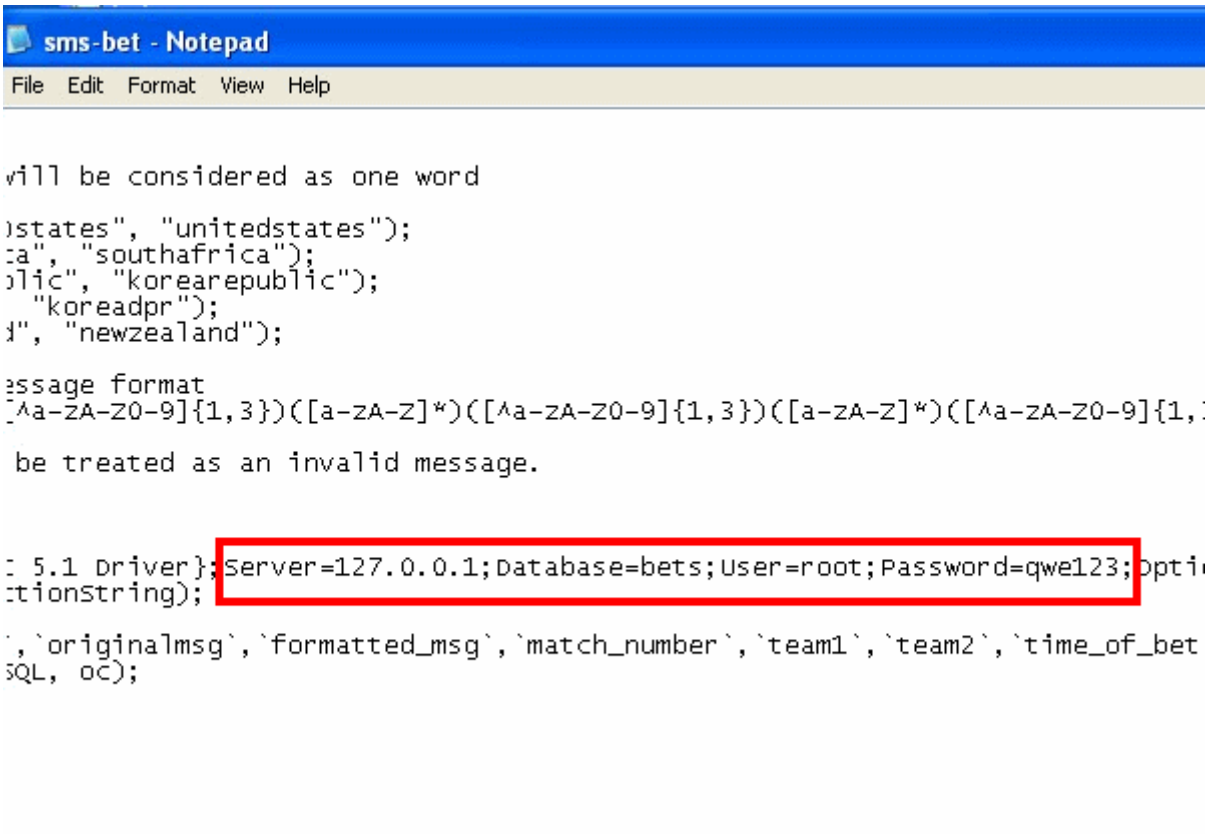


Figure 9 - Customize aspx file

sms-bet.aspx file

```
<%@ Page Language="C#"%>
<%@ Import Namespace="System" %>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.Common" %>
<%@ Import Namespace="System.Data.SqlClient" %>
<%@ Import Namespace="System.Data.Odbc" %>

<%

      // Copying the message parameters

   string sender = Request.QueryString["sender"];
   string messagedata = Request.QueryString["messagedata"];
   string messageid = Request.QueryString["messageid"];

      // Converting the message datra to lowercase

      string data = messagedata.ToLower();


      // The country names containing more parts will be considered as one word

      data = Regex.Replace(data, "united(.|..|...)states", "unitedstates");
   data = Regex.Replace(data, "south(.|..|...)africa", "southafrica");
   data = Regex.Replace(data, "korea(.|..|...)republic", "korearepublic");
   data = Regex.Replace(data, "korea(.|..|...)dpr", "koreadpr");
   data = Regex.Replace(data, "new(.|..|...)zealand", "newzealand");

      // Checking the match for the pre-defined message format
   bool match = Regex.IsMatch(data, "([0-9]{1,2})([^a-zA-Z0-9]{1,3})([a-zA-Z]*)
   ([^a-zA-Z0-9]{1,3})([a-zA-Z]*)([^a-zA-Z0-9]{1,3})([0-9]{1,3})([^a-zA-Z0-9]{1,3})([0-9]{1,3})");
```

```
    // If it not matches, then the message will be treated as an invalid message.
if (!match)
{
    bool connectedToDb;
    string connectionString = "Driver={MySQL ODBC 5.1 Driver};Server=127.0.0.1;
    Database=bets;User=root;Password=qwe123;Option=4;";
    OdbcConnection oc = new OdbcConnection(connectionString);
    oc.Open();
    string insertSQL = "insert into bets(`sender`,`originalmsg`,`formatted_msg`,
    `match_number`,`team1`,`team2`,`time_of_bet`,`msgid`) values ('" + sender + "','" +
    messagedata + "','invalid','0','invalid','invalid',now(),'" + messageid + "');";
    OdbcCommand command = new OdbcCommand(insertSQL, oc);
    command.ExecuteNonQuery();
    oc.Close();
}
else
{

// Replacing the special characters into semicolon.
    data = data.Replace('-', ';');
    data = data.Replace('.', ';');
    data = data.Replace(',', ';');
    data = data.Replace('_', ';');
    data = data.Replace('<', ';');
    data = data.Replace('>', ';');
    data = data.Replace('#', ';');
    data = data.Replace('&', ';');
    data = data.Replace('@', ';');
    data = data.Replace('{', ';');
    data = data.Replace('}', ';');
    data = data.Replace(':', ';');
    data = data.Replace('*', ';');
    data = data.Replace('\'', ';');
    data = data.Replace('\\', ';');
    data = data.Replace('"', ';');
    data = data.Replace('+', ';');
    data = data.Replace('!', ';');
    data = data.Replace('%', ';');
    data = data.Replace('/', ';');
    data = data.Replace('=', ';');
    data = data.Replace('|', ';');
    data = data.Replace('[', ';');
    data = data.Replace(']', ';');
    data = data.Replace('(', ';');
    data = data.Replace(')', ';');
    data = data.Replace('', ';');

    // Splitting the message by semicolons, and removing the incidental duplicate semicolons
    string[] Splitted = data.Split(';', ' ');
    string FormattedBet = "";
    for (int i = 0; i <= Splitted.Length - 1; i++)
    {
        if (Splitted[i] != "")
        {
            FormattedBet = FormattedBet + (Splitted[i] + ";");
        }

    }
```

```
    // Bet variable will contain the processed message, in an array
    string[] Bet = FormattedBet.Split(';');

    bool connectedToDb;

    try
    {
       string connectionString = "Driver={MySQL ODBC 5.1 Driver};Server=127.0.0.1;
       Database=bets;User=root;Password=qwe123;Option=4;";
       OdbcConnection oc = new OdbcConnection(connectionString);
       oc.Open();
       connectedToDb = true;
       try
       {

        // Ensuring the possibility for the team names could be swithable.

          string team1;
          string team2;
          if ((Bet[1].Substring(0, 3).CompareTo(Bet[2].Substring(0, 3))) < 0)
          {
             team1 = Bet[1].Trim() + ";" + Bet[3].Trim();
             team2 = Bet[2].Trim() + ";" + Bet[4].Trim();
          }
          else
          {
             team1 = Bet[2].Trim() + ";" + Bet[4].Trim();
             team2 = Bet[1].Trim() + ";" + Bet[3].Trim();
          }

        // Assembling the SQL query, and inserting the bet.
          string insertSQL = "insert into bets(`sender`,`originalmsg`,`formatted_msg`,
          `match_number`,`team1`,`team2`,`time_of_bet`,`msgid`) values ('" +
          sender + "','" + messagedata + "','" + FormattedBet + "','" + Bet[0].Trim()
          + "','" + team1 + "','" + team2 + "',now(),'"
          + messageid + "');";
          OdbcCommand command = new OdbcCommand(insertSQL, oc);
          command.ExecuteNonQuery();
          oc.Close();
       }
       catch (Exception z)
       {
          string errCode = z.Message;
          Response.Write(errCode);
       }

    }
    catch (Exception e)
    {
       string respmsg = "Cannot connect to database." + e.Message;
       connectedToDb = false;
       Response.Write(respmsg);
    }
  }
%>
```

Open query.php file (can be found in MySQL-table-structure.txt file). You need to specify the server, the user and its password for the MySQL function. In our example these parameters are the follows: localhost, root, and qwe123 in order.

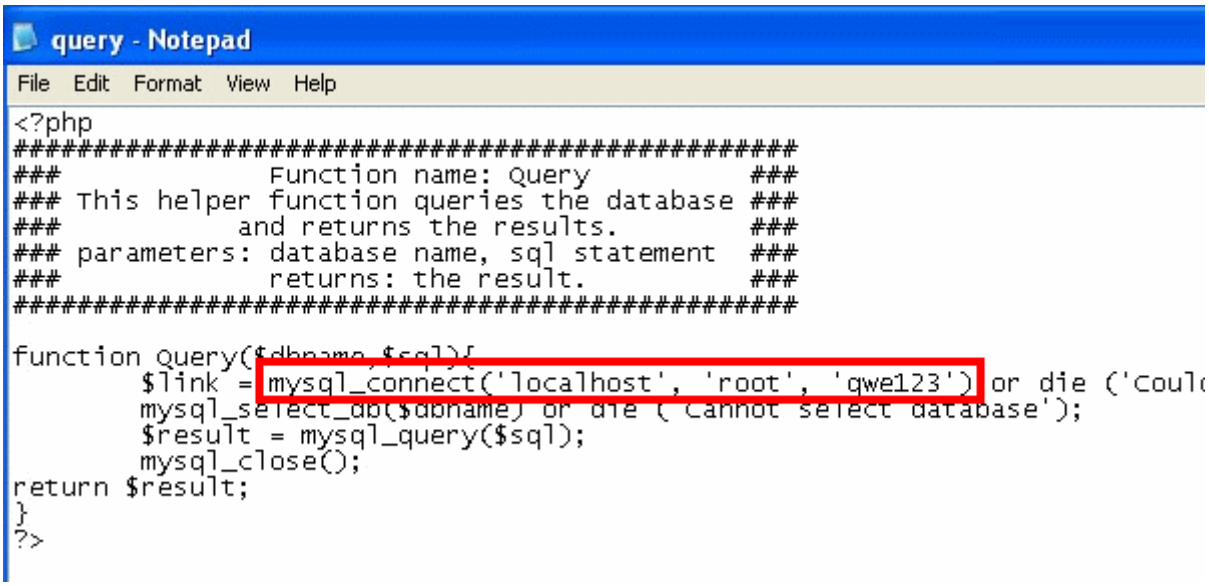


Figure 10 - Query.php file

query.php file

```
<?php
##################################################
###          Function name: Query          ###
### This helper function queries the database ###
###          and returns the results.       ###
### parameters: database name, sql statement  ###
###          returns: the result.           ###
##################################################

function Query($dbname,$sql){
      $link = mysql_connect('localhost', 'root', 'qwe123') or die
      ('Could not connect to MySQL database. Reason: ' . mysql_error());
      mysql_select_db($dbname) or die ('Cannot select database');
      $result = mysql_query($sql);
      mysql_close();
return $result;
}
?>
```

Test the system

I have sent a test message that has been processed by ASP user (Figure 11). This was a test message and not a bet.

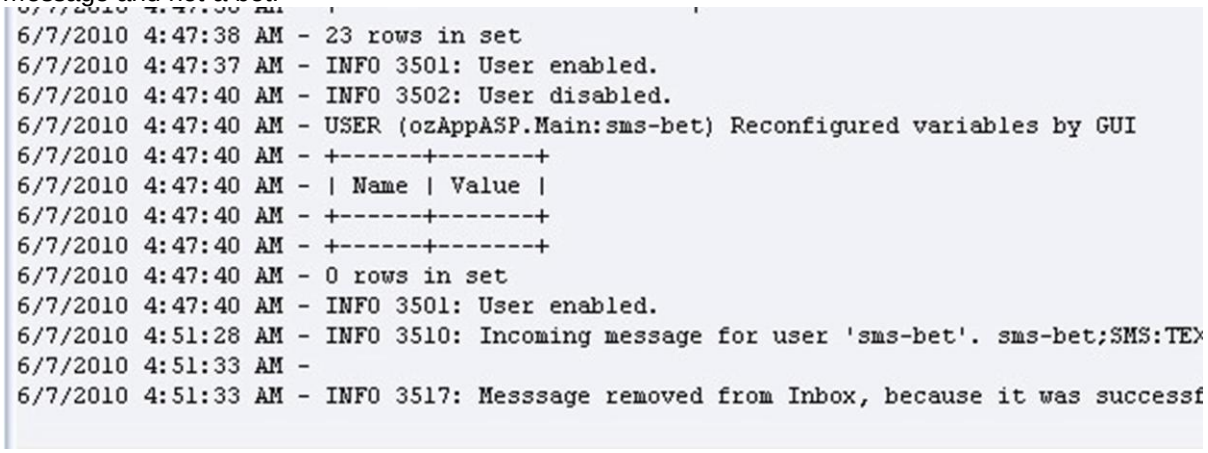


Figure 11 - Test message processed by ASP user

To see the results, click on "view results" on the control pane on the web site (Figure 12).

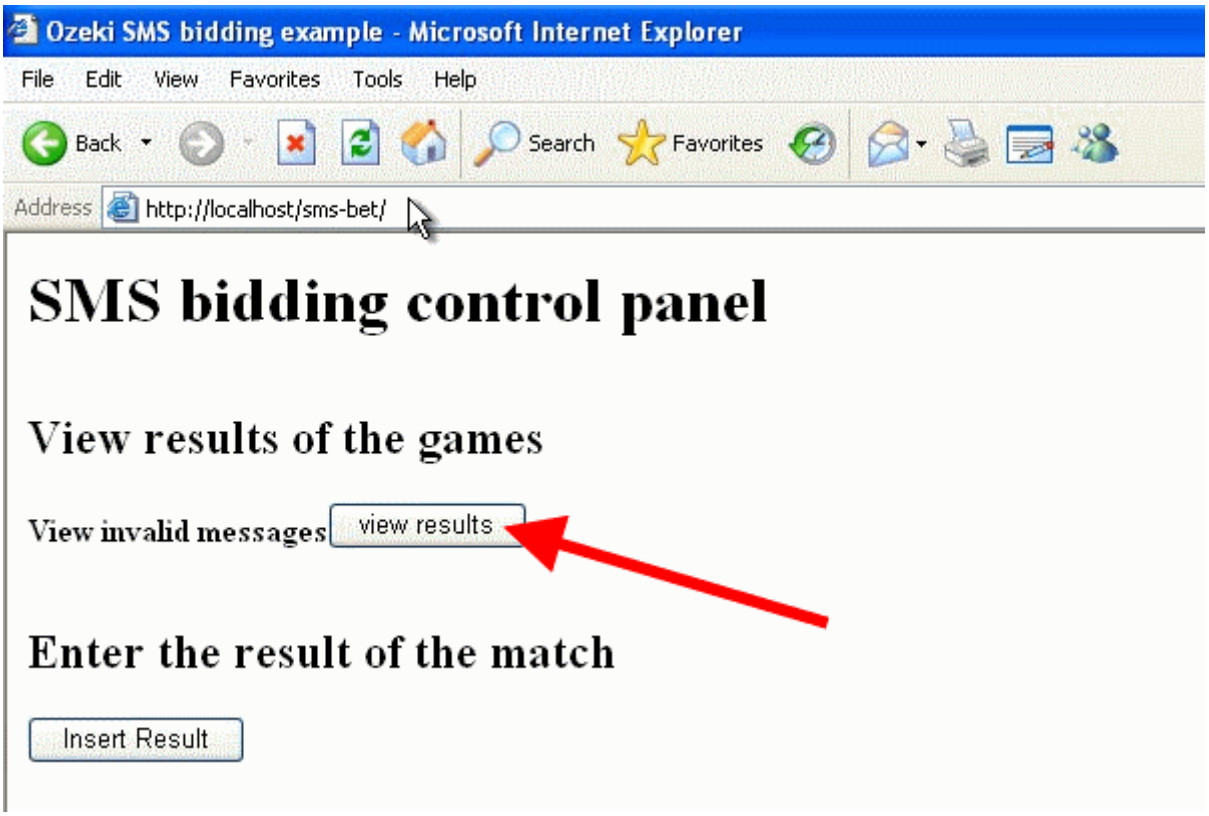


Figure 12 - View results

Since my first test message was not a bet, it is considered by the system as an invalid message (Figure 13).

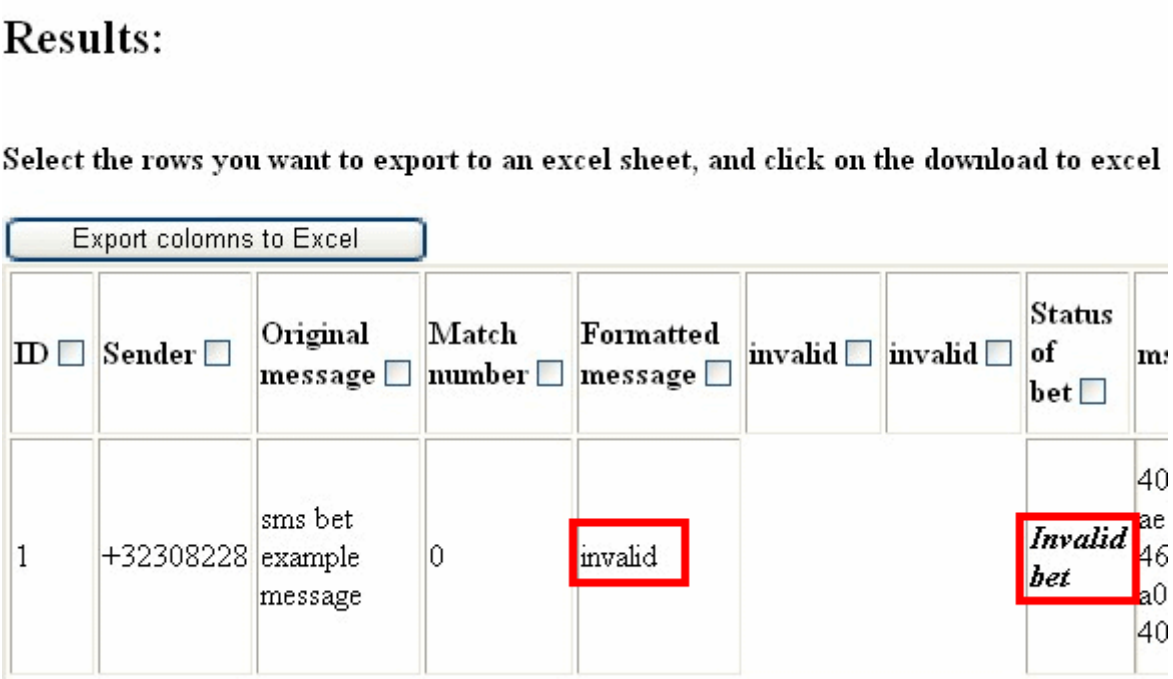


Figure 13 - Invalid message

Create a new game for betting

First you can enable or disable restrictions to the system. If you enable it, it means that the system will accept only one bet from one user during one game period. If you disable this option then users can send several bets and these bets will be accepted by the system (if they meet the requirements) (Figure 14).

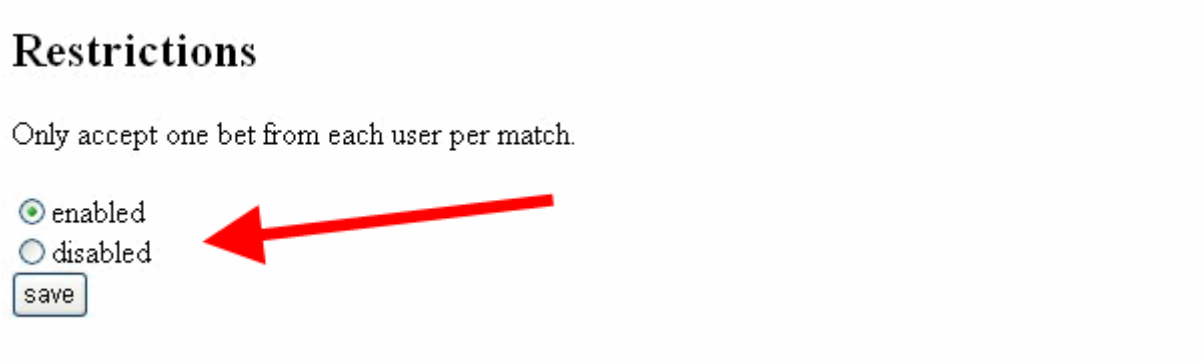


Figure 14 - Enable/Disable restrictions

In Create game for the next match section I create a test game:

Ordinal number of the match: it is necessary to specify an identification number for the game (in this example it is "2"). Users need to include this number in their SMS bets, without this parameter the bit will be "invalid"

Team1: Specify the name of the first team (France)

Team2: Specify the name of the other team (Uruguay)

Match begin time: Specify the exact date of the match in Year/Month/Day Hour:Minute:Second format (2010/06/11 20:30:00)

Game period: It is the period in which users can send bets and the system will accept them if they meet the requirements. You need to specify a from - to period. If you do not specify Hour/Minute/Second then the system will count the period from Hour 00/00/00. So in this example this period is 2010/06/06 - 2010/06/10, it means that bets are accepted from 2010/06/06 00/00/00 to 2010/06/10 00/00/00

To add the new created game click on Create new game (Figure 15).

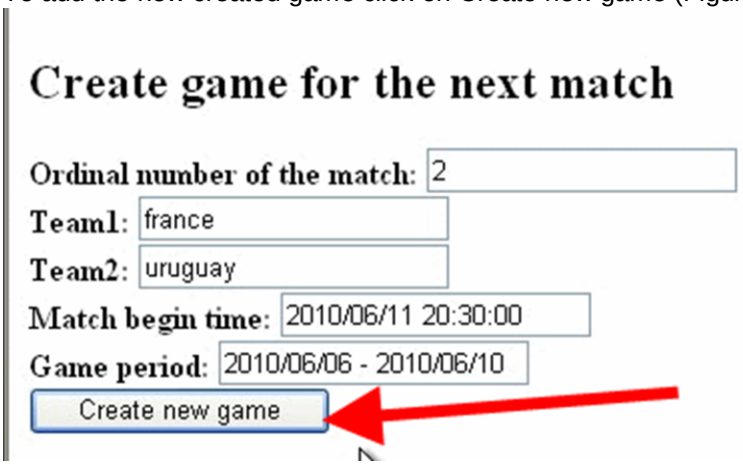


Figure 15 - Create new game

Now I send a test bet. In the SMS text message the follows need to be included: Ordinal number of the match, Team1, Team2, Result of Team1, Result of Team2. Please note that the system is flexible since it is able to handle cases if users invert the names of teams (it selects only the names and the order does not matter), however, the names needs to be spelled correctly. Furthermore, the system picks unnecessary punctuation marks up to 3 characters between the values in the message. It means that users can write any punctuation mark in their messages, the system will ignore them up to 3 characters.

If all of the requirements are fulfilled, the system accepts the bet. If something is missing, or the user misspells the country names the message will be inserted as an invalid message. In case of accepted bets, the bet will be inserted into the table of the respective game which the bet arrived at. The following information is provided (Figure 16):

ID: this helps identify the bet in the database
Sender: the phone number of the sender
Original message: shows how the message was received
Match number: identifies the game
Formatted message: it is the accepted message. It shows the form in which the message has been processed
Team1: (France) the name of the first team
Team2: (Uruguay) the name of the other team
Status of bet: it shows whether the bet is valid or invalid
msgID: this is the message ID that helps identify the message in Ozeki NG SMS Gateway
Time of the match: the starting time of the match
Interval of the bidding: it shows the game period in which bets are accepted
Time of the bet: it is the time when the bet is received in the system
Status according to time: it shows if the message arrives in time, before the betting period or after the betting period
Match result: the final results of the match
Won/Lost: it shows that the respective user wins or loses with his bet

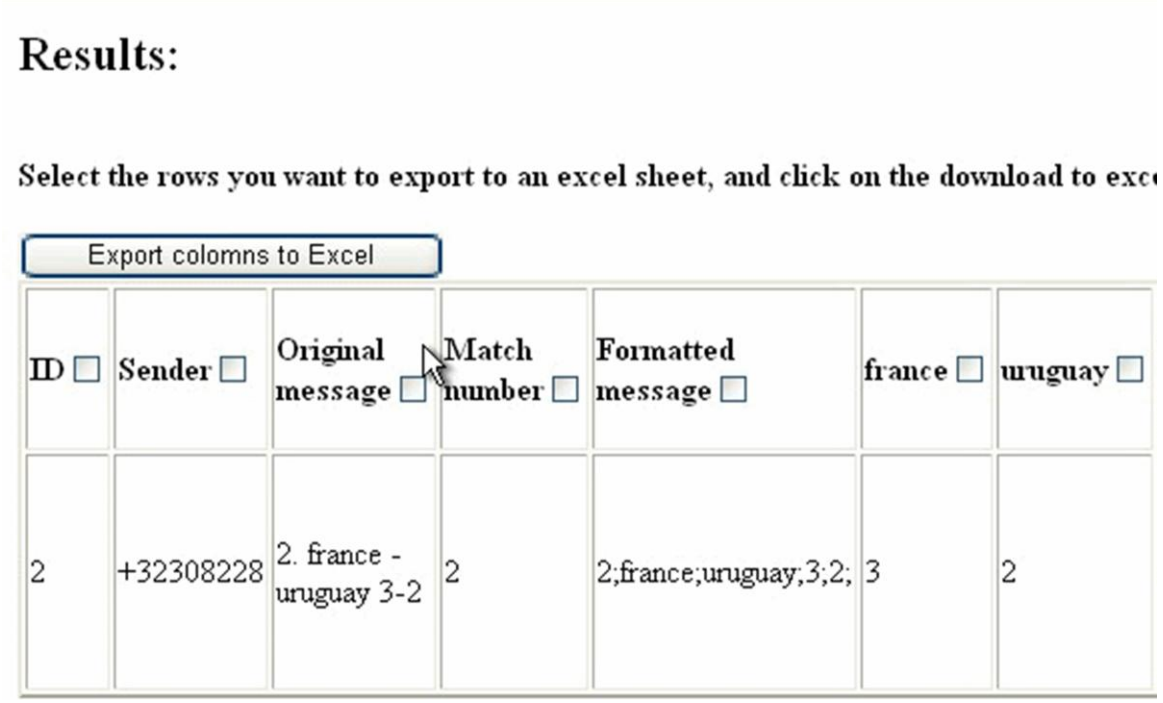


Figure 16 - Test bet

Now I enter a test result for the match in Enter the result of the match section. I specify the result (france-uruguay 3-2) and click on Insert result (Figure 17).

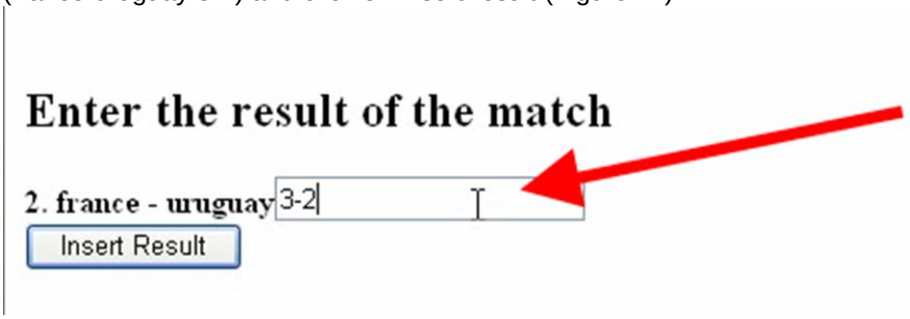


Figure 17 - Insert result

Since the test bet meets all the requirements (even the result of the game is agreed) Figure 18 demonstrates that it won the game.

Status of bet	msgID	Time of the match	Interval of the bidding	Time of the bet	Status according to time	Match result	Won/Lost
valid	86da5d2d-356c-4c9e-8e68-072aa0	June 11, 2010, 8:30 pm	June 6, 2010, 12:00 am - June 10, 2010, 12:00 am	2010-06-07 04:53:10	Arrived on time	france: 3 uruguay: 2	won

Figure 18 - Won the game

Finally, it is possible to select columns and by clicking on Export columns to Excel button, the selected columns will be exported into an Excel file (Figure 19).

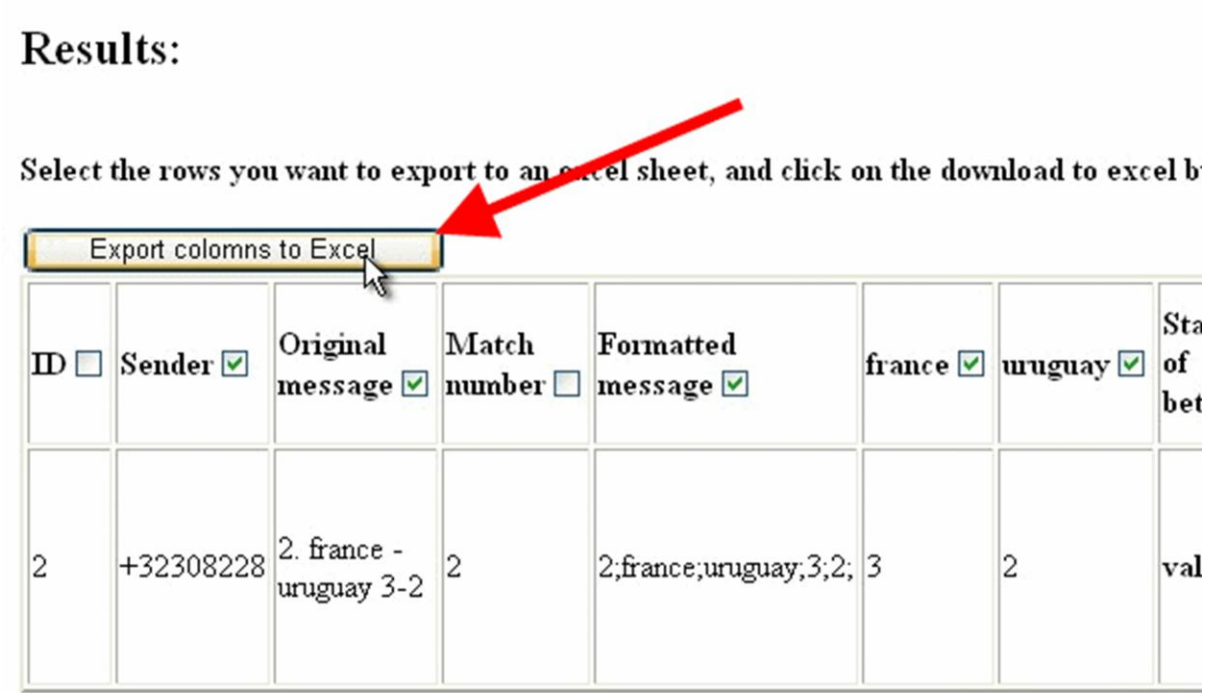


Figure 19 - Export columns to Excel

On Figure 20 you can see the exported columns in Excel.

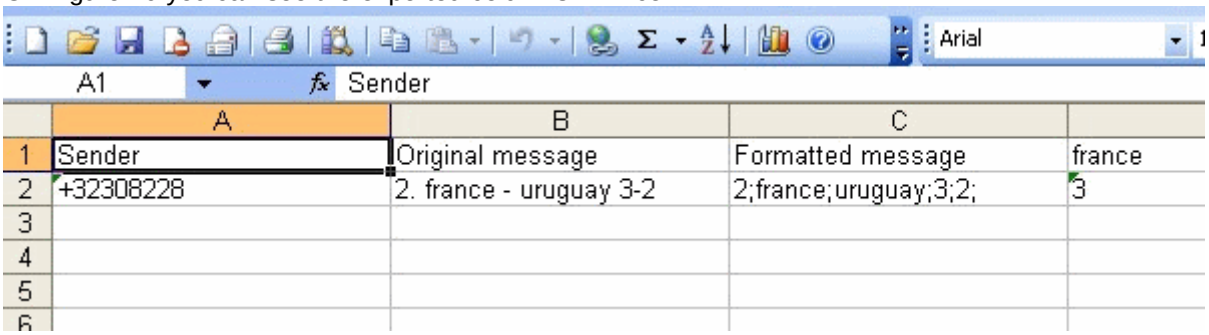


Figure 20 - Exported columns in Excel

If you haven't downloaded Ozeki NG SMS Gateway, download the software or get a free trial right now: Download Ozeki NG SMS Gateway

Birthday greeting SMS service with Oracle, Ozeki NG SMS Gateway software and QuesCom GSM Gateway

This page provides you detailed setup instructions on how to send birthday SMS greetings from Oracle database with Ozeki NG SMS Gateway through QuesCom GSM gateway.

First of all you need to configure Ozeki NG SMS Gateway software to be able to send SMS messages from your PC. You can find step-by-step guide at: Quick start guide. The Ozeki SMS software connects to a QuesCom GSM gateway.

After you have configured your Ozeki NG SMS Gateway software and QuesCom GSM Gateway, you need to configure your Oracle database. Please find the configuration guide at: How to send SMS from Oracle.

If you have configured your system you can start to setup the birthday greeting SMS service.

Configuration guide

So it is assumed that you have already installed QuesCom GSM gateway, Ozeki NG SMS Gateway software and Oracle database with all its components.

These are the main steps of the configuration of birthday greeting SMS service:

Start SQL Developer (it is a component of Oracle database)

Create the connection to database

Create database table

Create a sequence to move the ID

Create a trigger to move the ID

Load data into the database

Install and configure a Database user in Ozeki NG SMS Gateway

To configure birthday greeting SMS service, start SQL Developer program of Oracle database in Start menu (Figure 1).

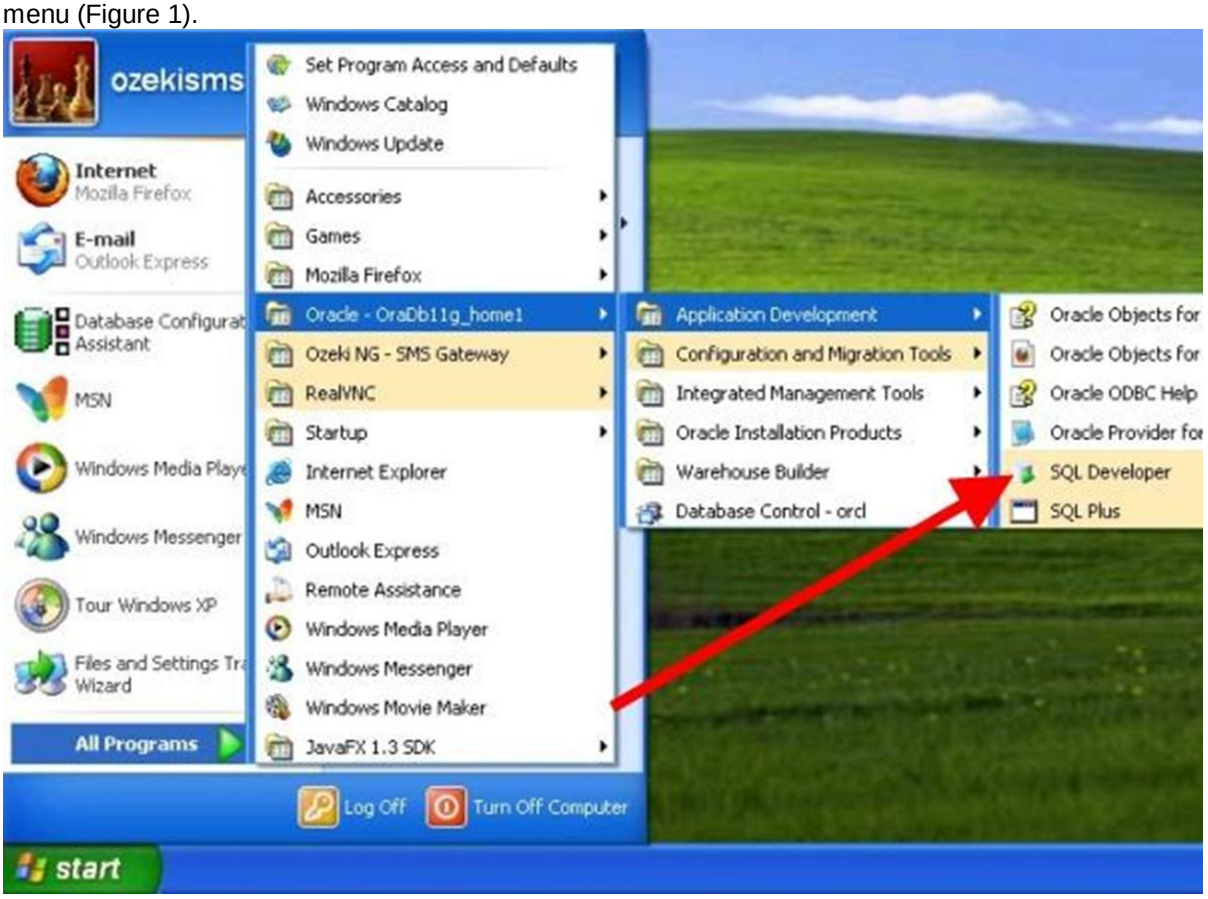


Figure 1 - Start SQL Developer

Add a new connection to the database by right clicking on Connections and click on New Connection (Figure 2).

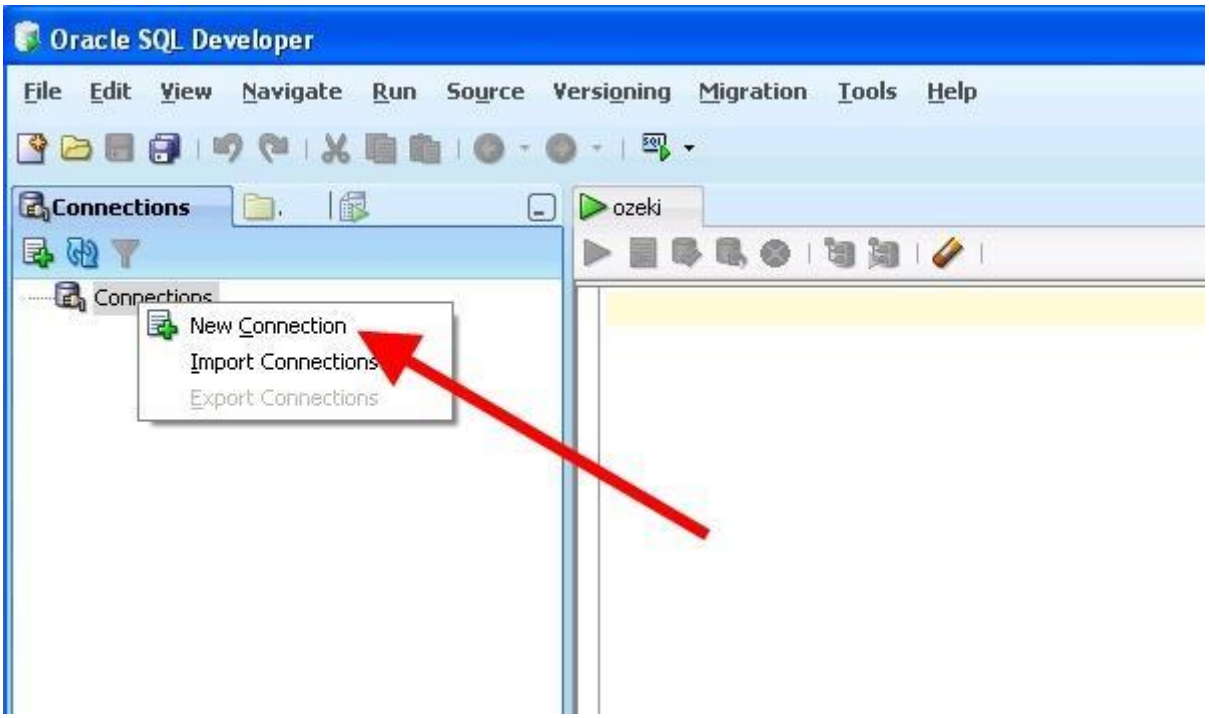


Figure 2 - Add new connection

Specify the parameters in the appeared window (Figure 3):
Connection Name: You can provide any name you wish (in our example it is "ozeki connection").
Username: the username you use to access to the database
Password: the password you use to access to the database

In SID field you need to enter the name that has been provided to the database when it is installed (installation name of the database).
Finally click on Connect.

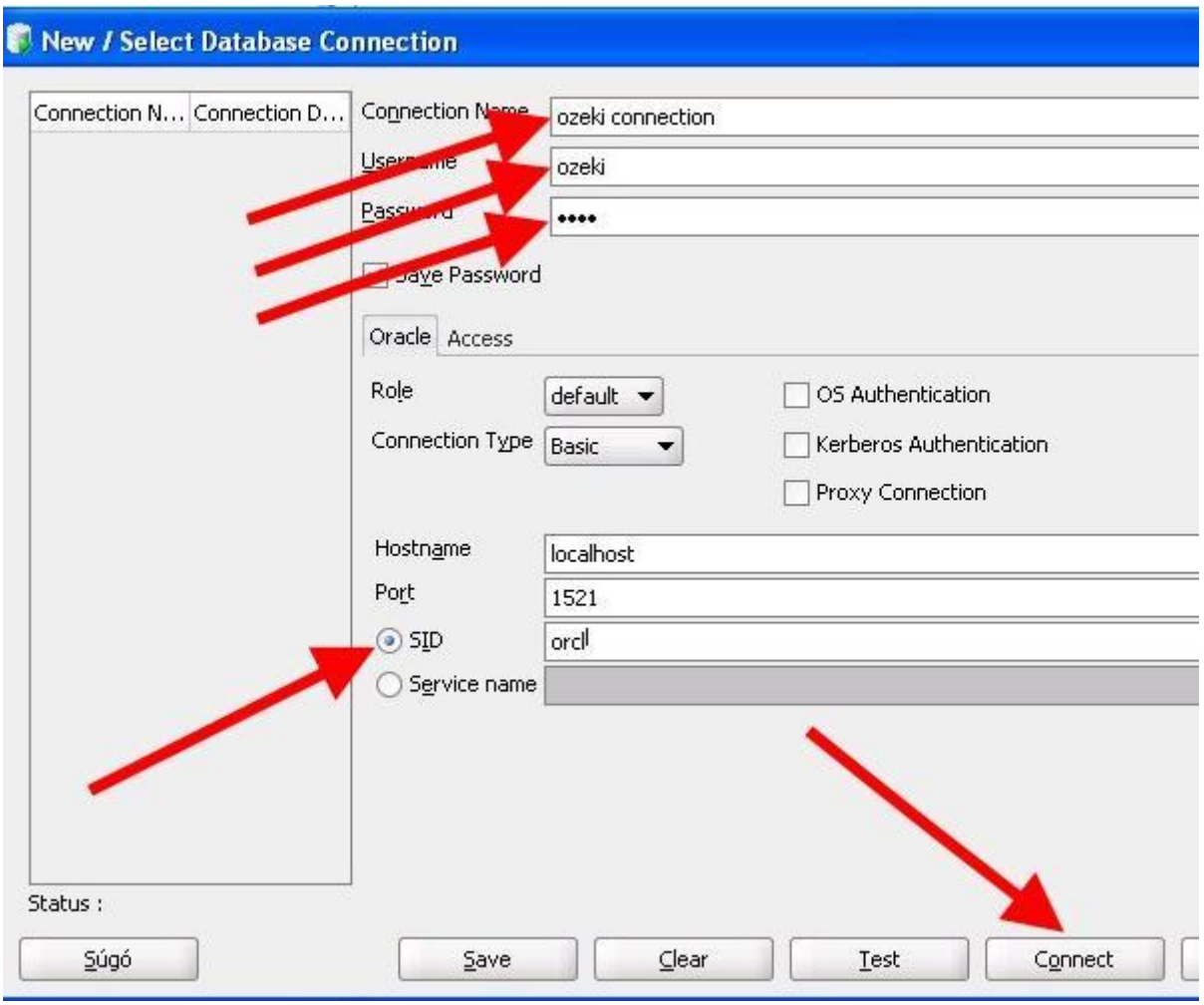


Figure 3 - Specify parameters for connection

On Figure 4 you can see that the connection has been created.

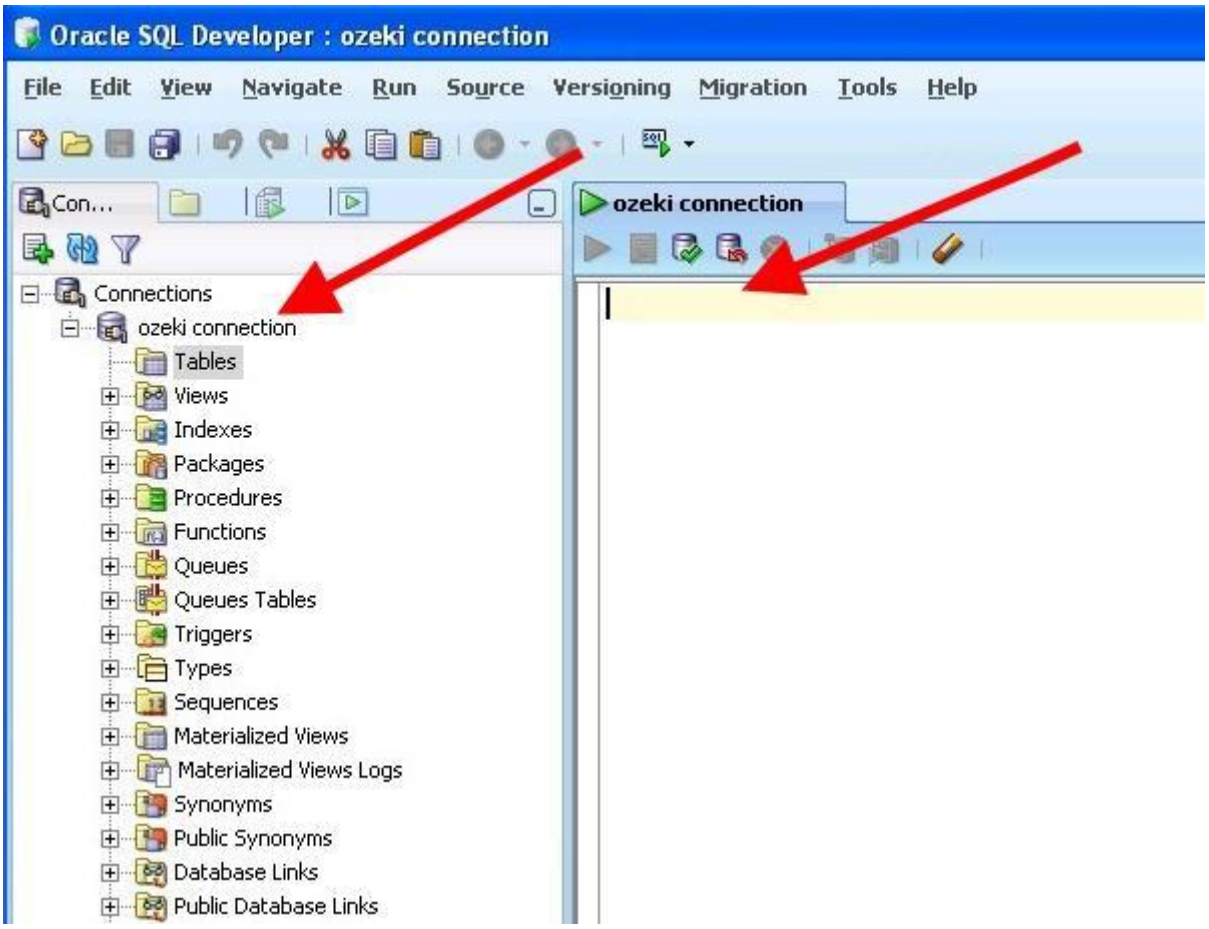


Figure 4 - Created connection

Now create a database table. In the empty field you can type the SQL statements that will create the required database table (Figure 5). To execute the SQL query click on the green arrow.

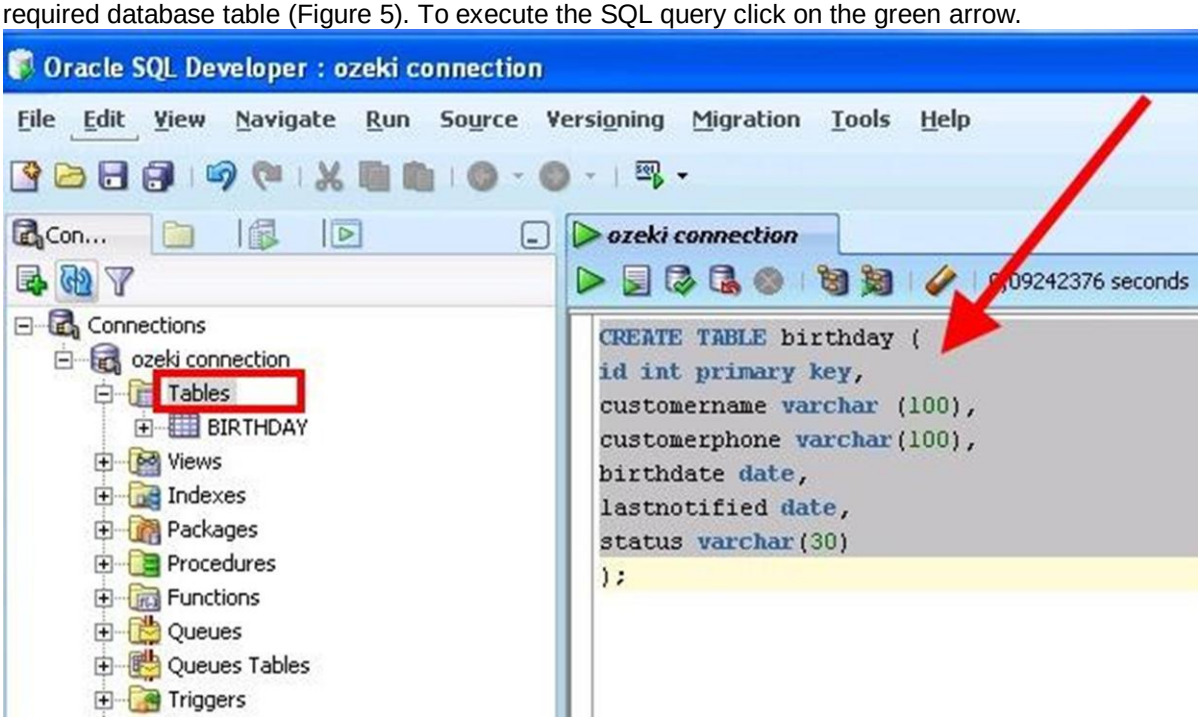


Figure 5 - SQL statement

On Figure 6 you can see the created database table.

ozeki connection | BIRTHDAY

Columns | Data | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Indexes | SQL

Actions...

Column Name	Data Type	Nullable	Data Default	COLUMN ID	Primary Key	COMMENTS
ID	NUMBER	No	(null)	1	1	(null)
CUSTOMERNAME	VARCHAR2(100 BY...	Yes	(null)	2	(null)	(null)
CUSTOMERPHONE	VARCHAR2(100 BY...	Yes	(null)	3	(null)	(null)
BIRTHDATE	DATE	Yes	(null)	4	(null)	(null)
LASTNOTIFIED	DATE	Yes	(null)	5	(null)	(null)
STATUS	VARCHAR2(30 BYTE)	Yes	(null)	6	(null)	(null)

Figure 6 - Created table

Create a sequence to move the primary key (ID) of the table (Figure 7).

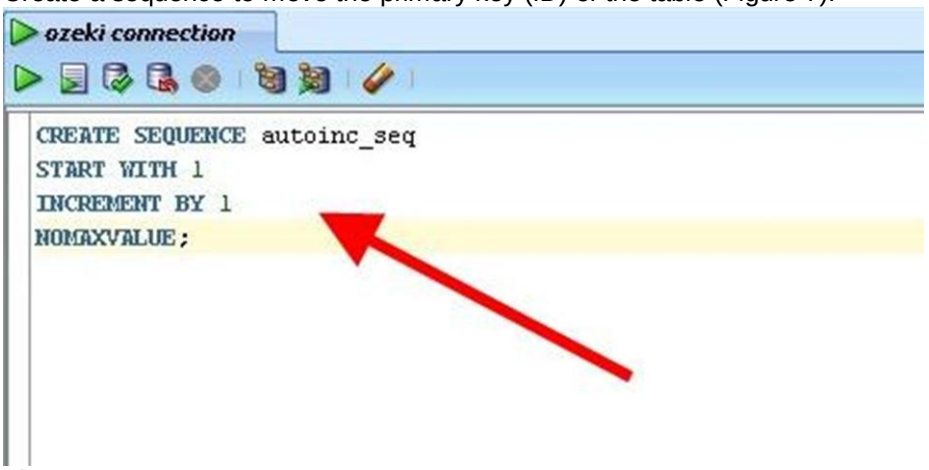


Figure 7 - Create a sequence

On Figure 8 you can see the created sequence.

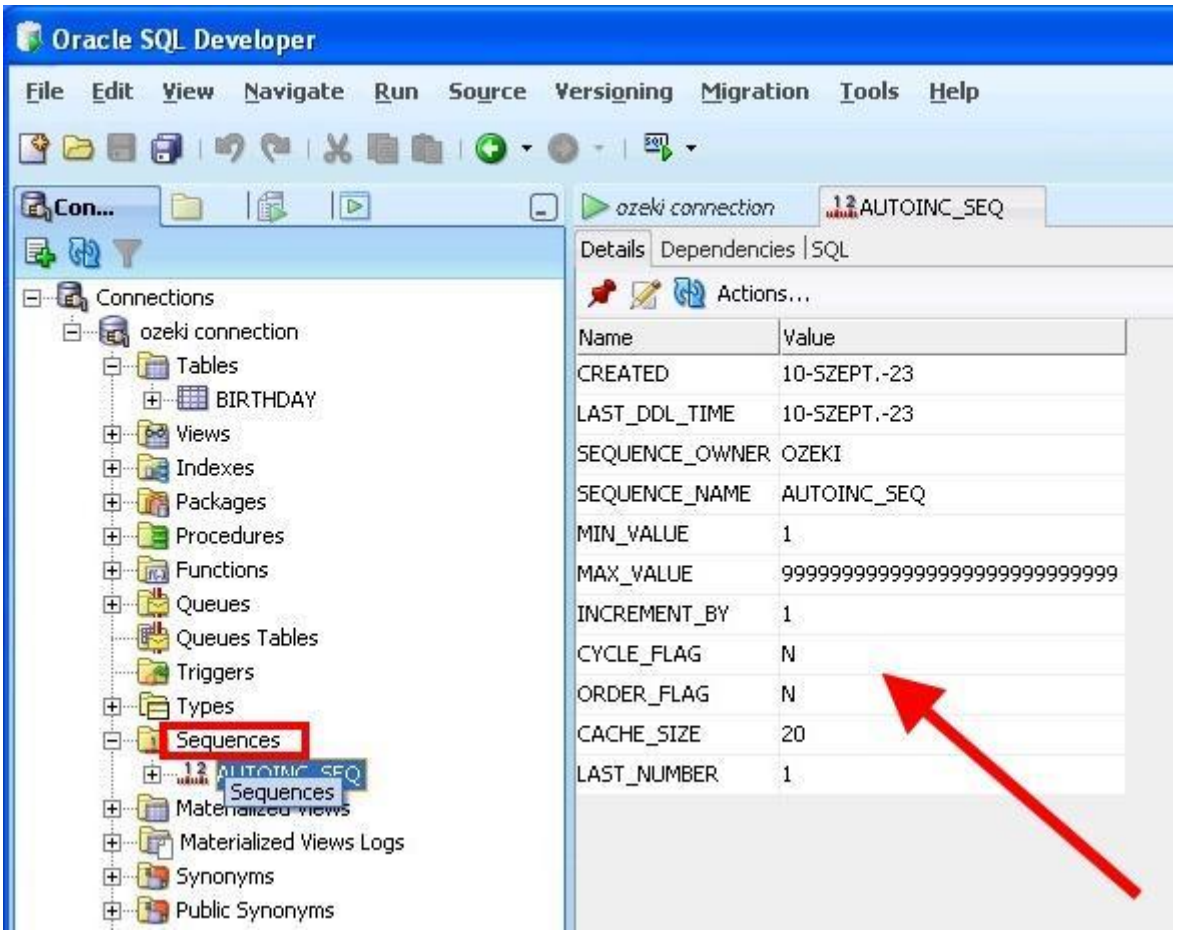


Figure 8 - Created sequence

You also need to create a trigger (Figure 9). This trigger uses the created sequence and moves the primary key (ID) of the table.

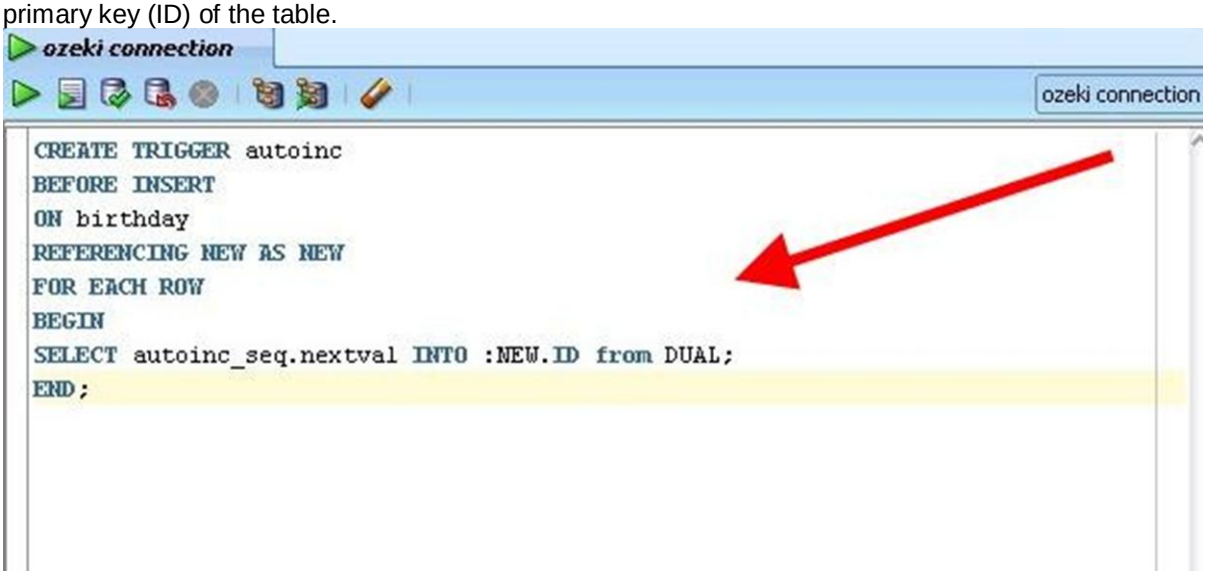


Figure 9 - Create a trigger

You can see the created trigger (Figure 10).

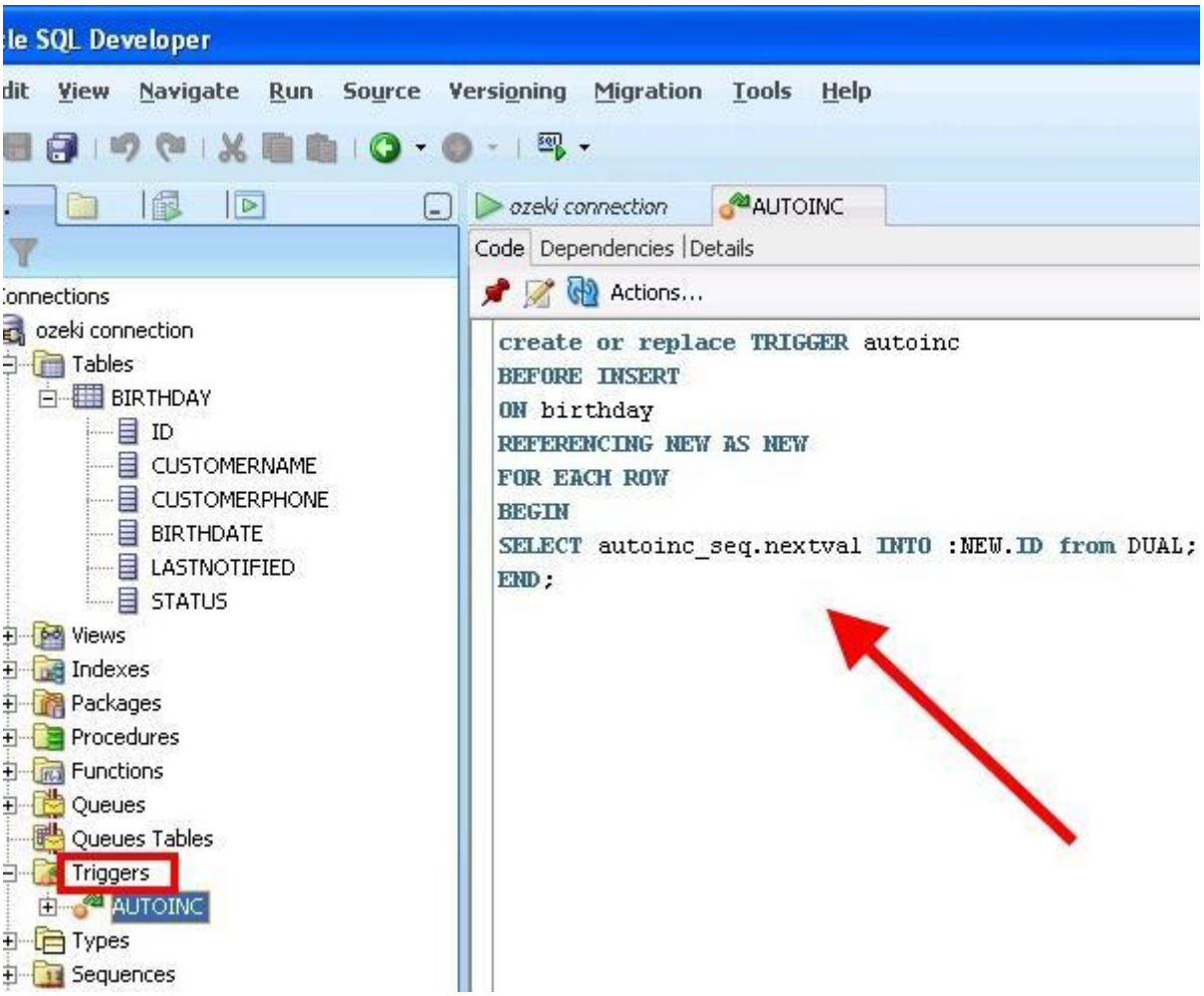


Figure 10 - Created trigger

On Figure 11 you can also see the created trigger among the triggers of birthday table.

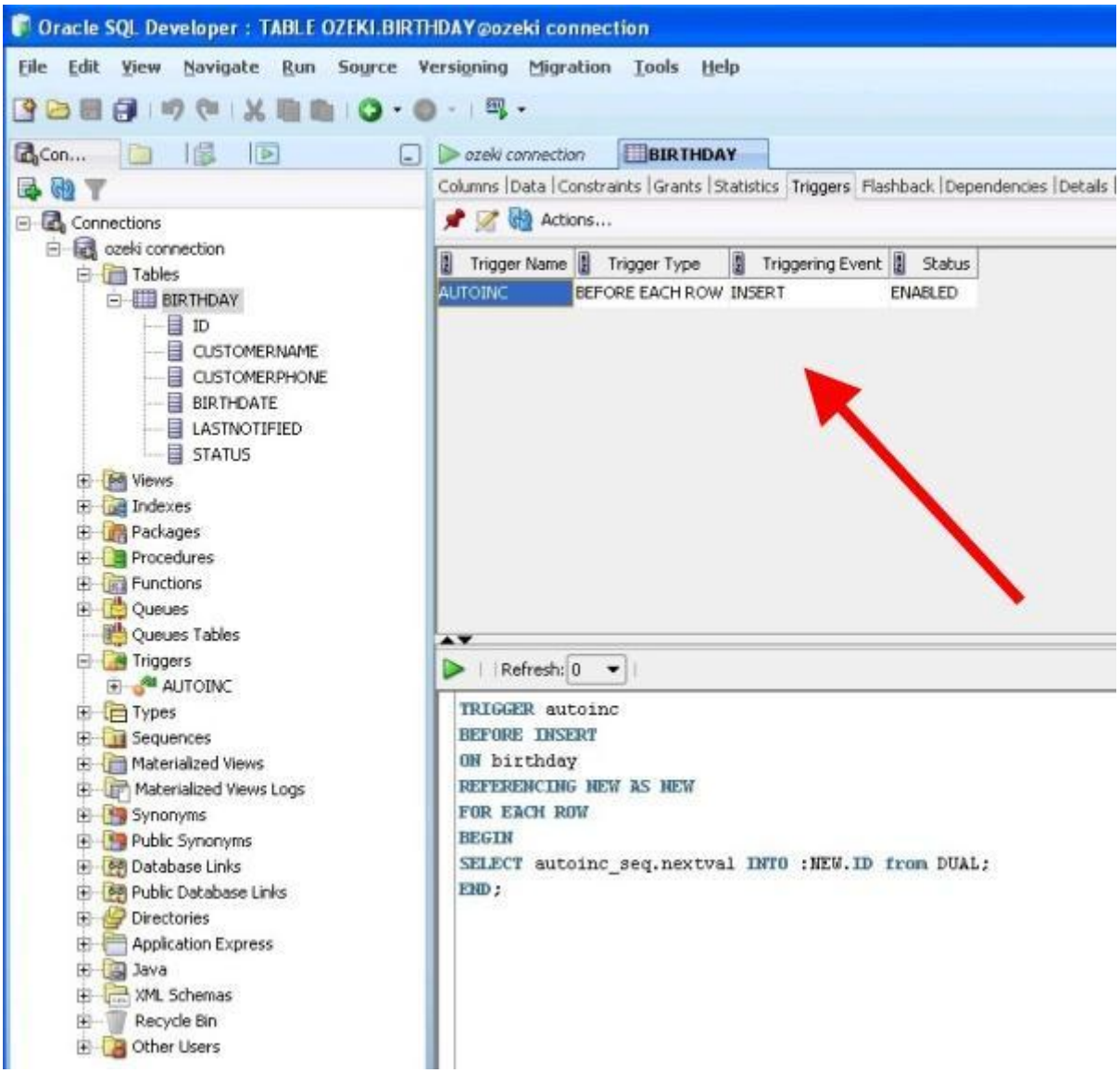


Figure 11 - Created trigger among the triggers of birthday table

Now you can enter data into the table (Figure 12) in the following way:

```
INSERT INTO birthday (customername, customerphone, birthdate) VALUES
('Elizabeth','+36301234567',DATE'2010-09-22');
```

Figure 12 - Enter data

Start Ozeki NG SMS Gateway service (Figure 13).

Figure 13 - Start Ozeki NG SMS Gateway

Log into Ozeki NG SMS Gateway (Figure 14).

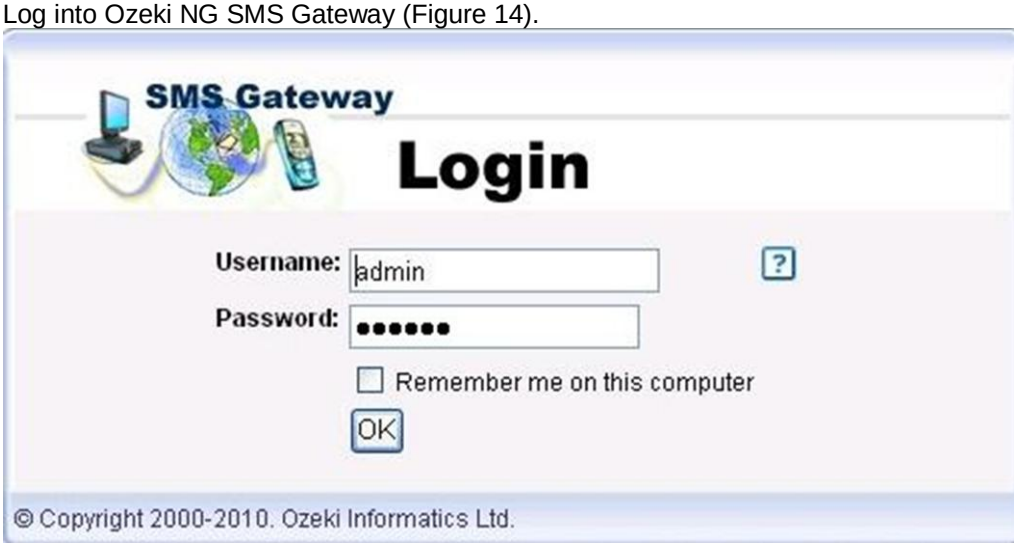


Figure 14 - Login

Install a Database user by clicking on Add new user or application (Figure 15).

Figure 15 - Add new user or application

Select Database user interface and click on Install next to it (Figure 16).

Figure 16 - Install Database user

Provide a name for the Database user - in our example it is "birthday_dbuser" (Figure 17).

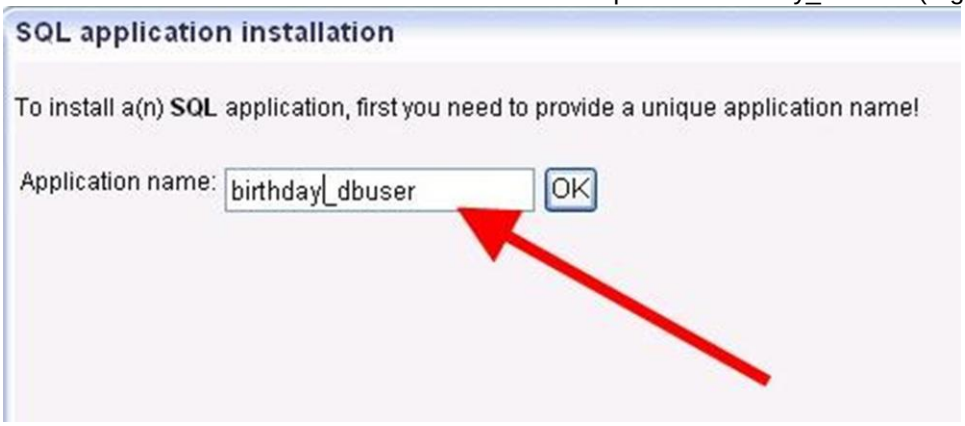


Figure 17 - Provide a name

Finally configure the installed Database user. On Database connection tab you need to specify the follows:

Connection string type: OleDb

Connection string:

```
Provider=OraOLEDB.Oracle.1;Persist Security Info=False;User ID=ozeki;Password=root;
```

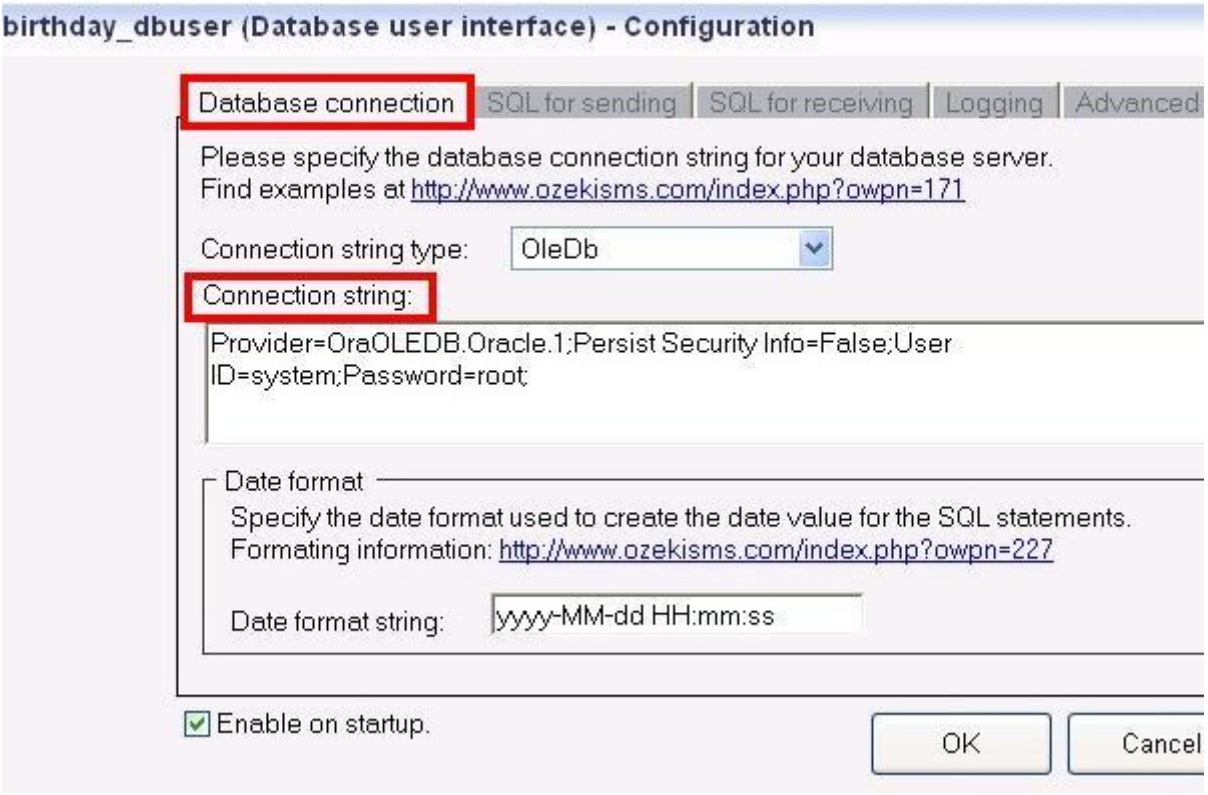


Figure 18 - Configuration of Database user

On SQL for sending tab you also need to specify the SQL statements.

Polling tab on SQL for sending panel:

```
SELECT id, '', customerphone, CONCAT(CONCAT(CONCAT(CONCAT('Hello ',customername),
'! The day of your birthday is: '), birthdate), ' Congratulations!')
FROM birthday
WHERE (EXTRACT(MONTH FROM CURRENT_DATE)=EXTRACT(MONTH FROM birthdate))
AND (EXTRACT(DAY FROM CURRENT_DATE)=EXTRACT(DAY FROM birthdate))
AND ((lastnotified is null) or (not(EXTRACT(YEAR FROM CURRENT_DATE))=(EXTRACT(YEAR
FROM lastnotified))))
```

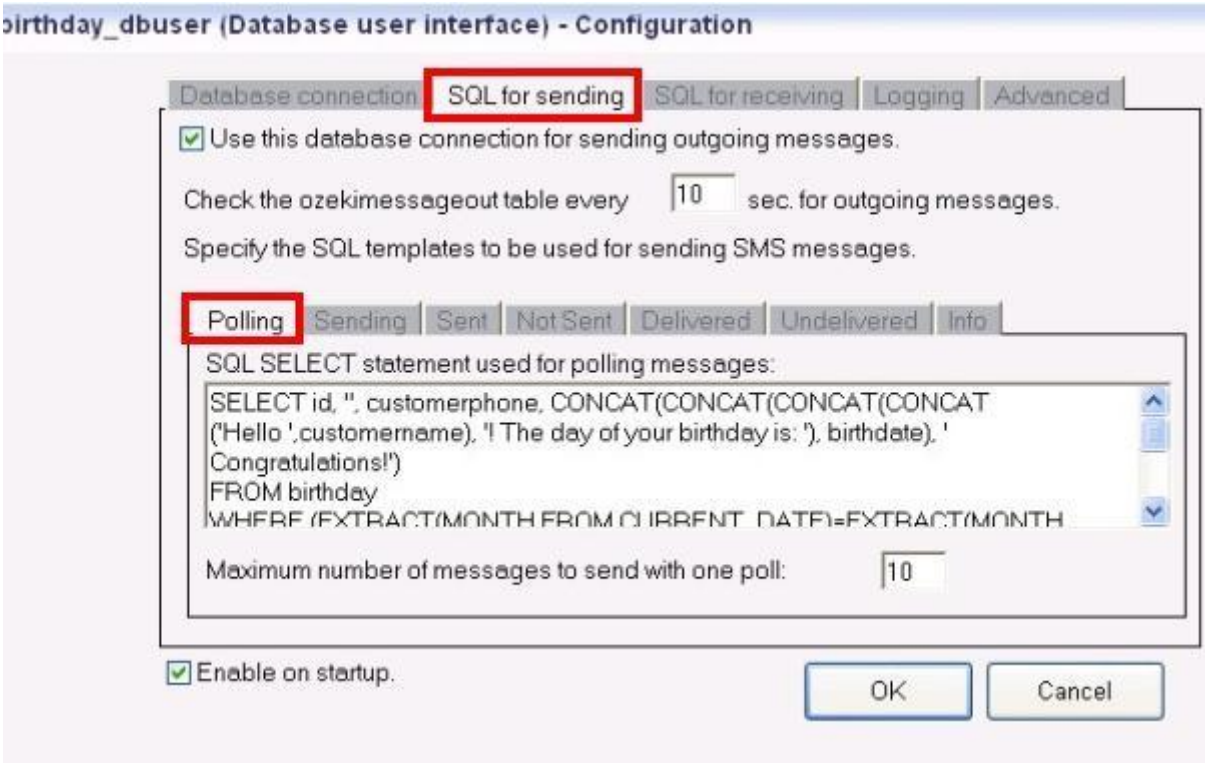


Figure 19 - Polling tab

Sending tab on SQL for sending panel:

```
UPDATE birthday SET status='sending', lastnotified=current_date WHERE id='$id'
```

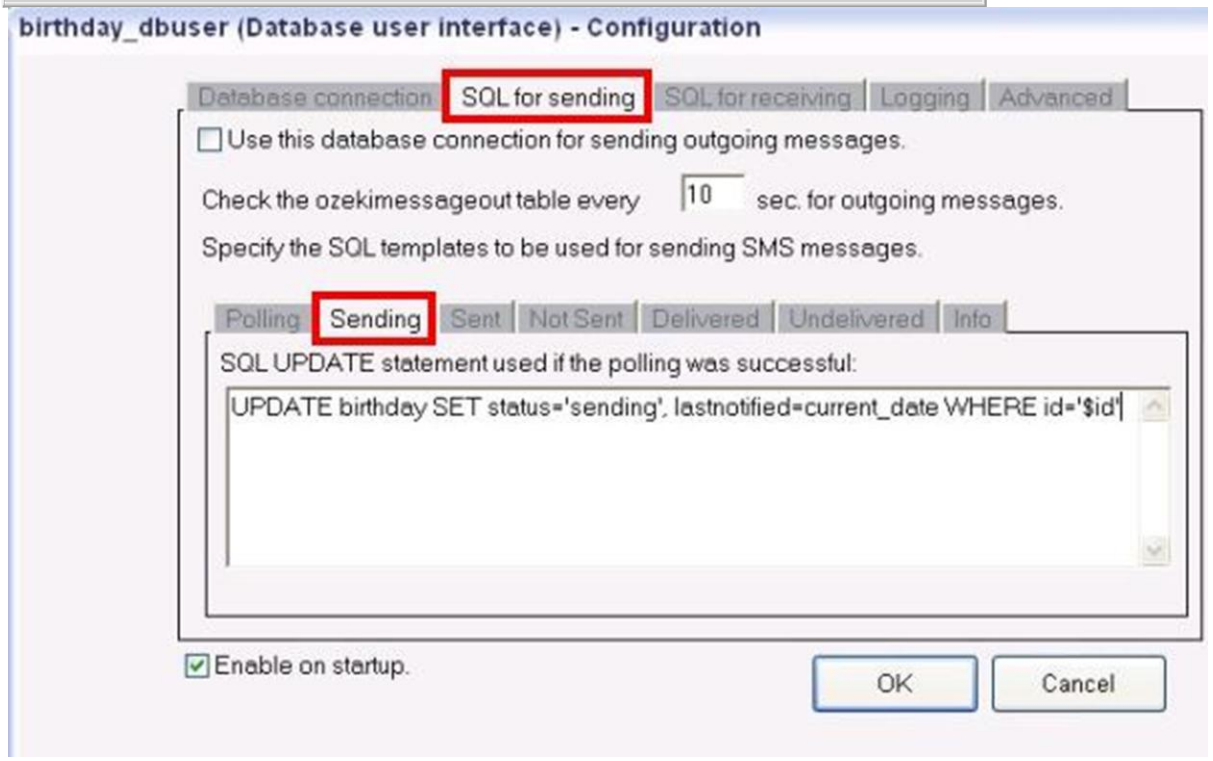


Figure 20 - Sending tab

Sent tab on SQL for sending panel:

```
UPDATE birthday SET status='sent', lastnotified=current_date WHERE id='$id'
```

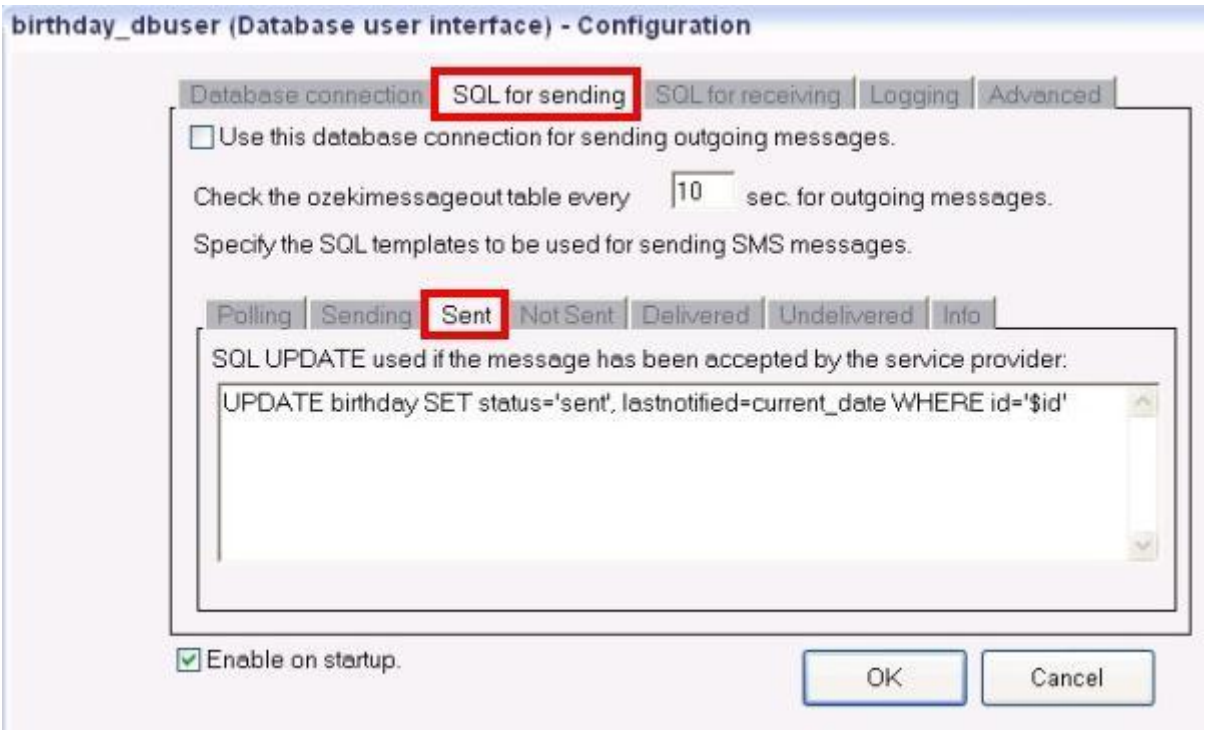


Figure 21 - Sent tab

Notsent tab on SQL for sending panel:

```
UPDATE birthday SET status='notsent', lastnotified=current_date WHERE id='$id'
```

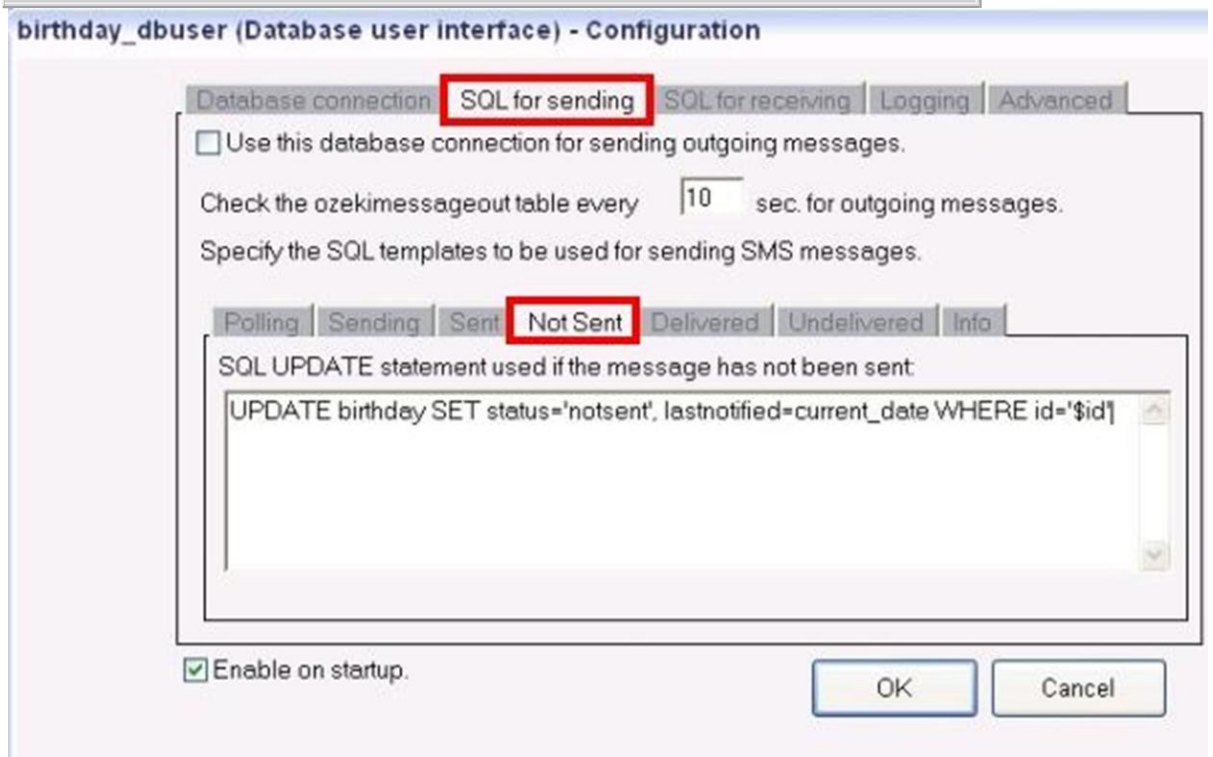


Figure 22 - Not sent tab

Delivered tab on SQL for sending panel:

```
UPDATE birthday SET status='delivered', lastnotified=current_date WHERE id='$id'
```

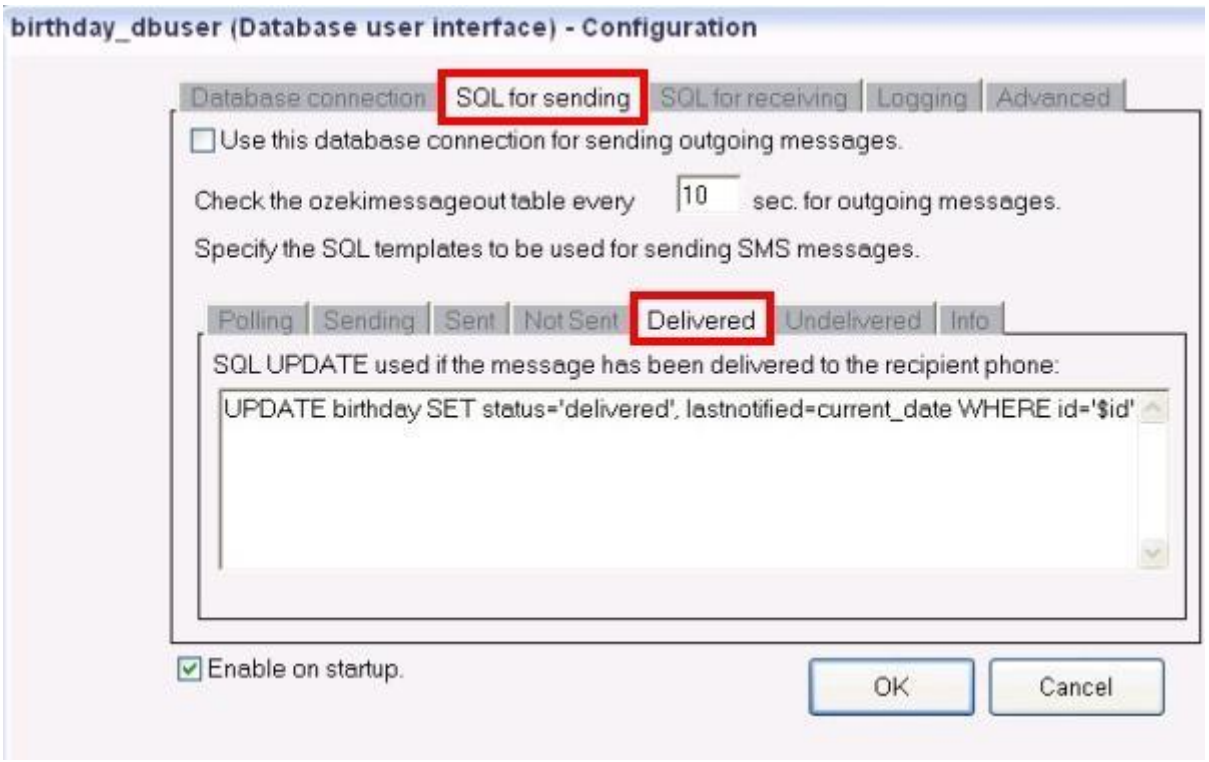


Figure 23 - Delivered

Undelivered tab on SQL for sending panel:

```
UPDATE birthday SET status='undelivered', lastnotified=current_date WHERE id='$id'
```

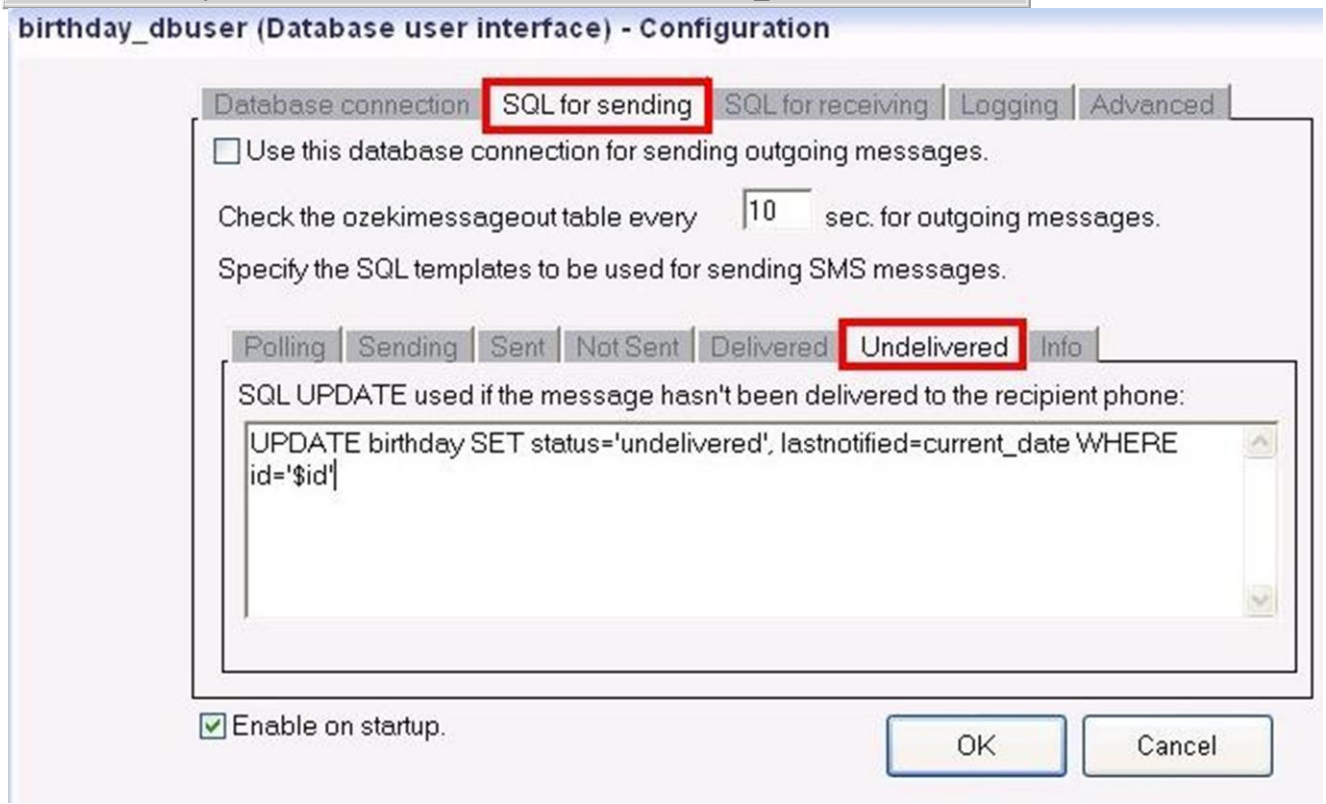


Figure 24 - Undelivered

On Figure 25 you can see that Ozeki NG SMS Gateway has sent out the test messages after I followed the configuration steps above.

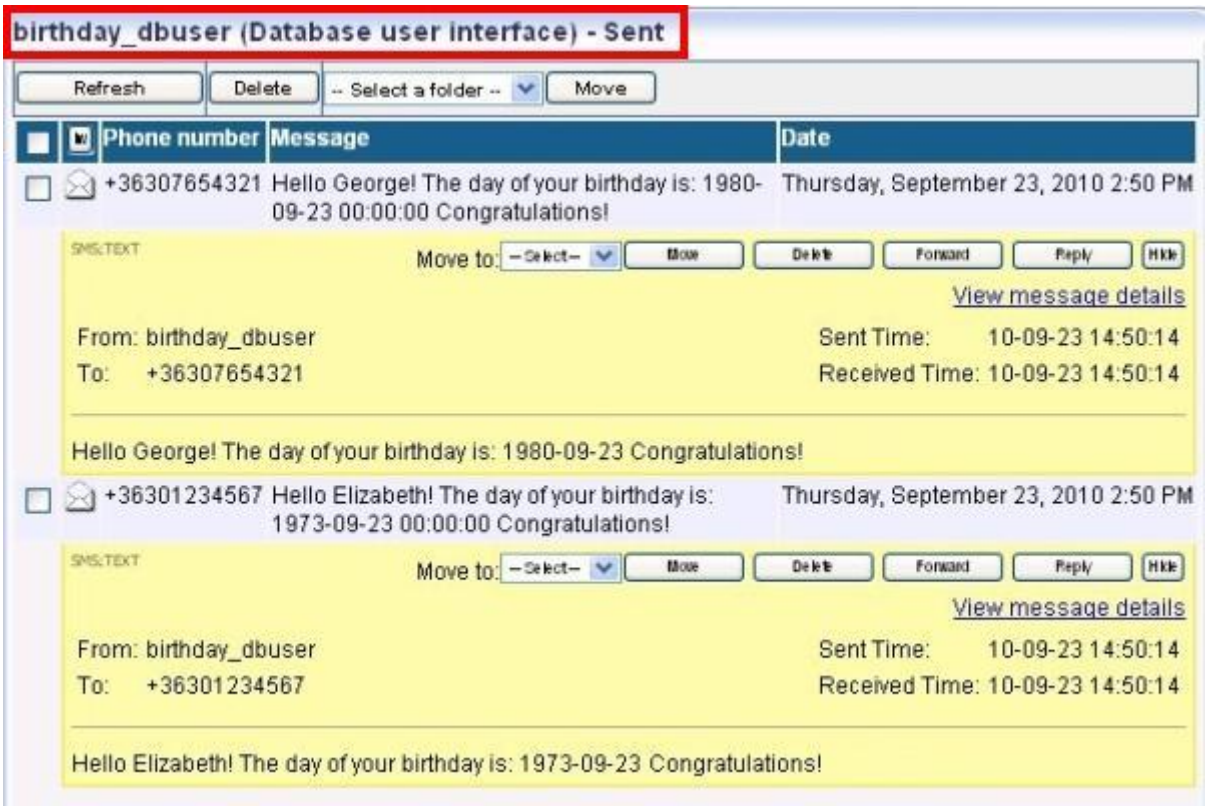


Figure 25 - Sent messages in Ozeki NG SMS Gateway

On Figure 26 you can also see that the status of the birthday greetings has also been modified to "sent" in the database.

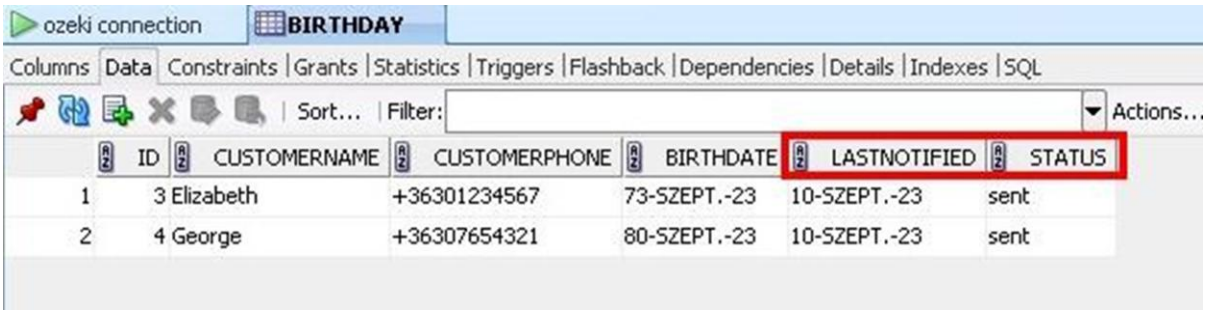


Figure 26 - Status is sent

10. Setup QuesCom GSM gateway and Ozeki link

WARNING minimum version is IAD06.50

1/ On Ozeki setup:

Create a new SMPP provider and go in "Configure" menu,
Under SMPP Settings, configure:
Host Ip of the Gateway,
TCP port 2775,
user and password of your Gateway (administrator/quescom by default).

Under "Message Data" tab, configure:
Prefered charset > ISO_8859_1
Policy > Enforce
Uncheck "Convert incoming GSM 7 to iso8859-1"

Save (OK)
"Disconnect" this new SMPP provider then "Connect" it.

Check in "Events" that you've got:
Successful login QuesCom
Connection Online.

2/ On QuesCom Gateway Web
Setup add new CTI application into QPortal, the name is “QSmsc”, respect capital letters for QS !

Then you have to add 2 Services into QPortal;
one for outgoing SMS (QSmsc -> Sms To Send)
and another for incoming SMS (SMS Incoming -> QSmsc)

Add "Phone number" for each SIM card under "SIM pool" interface

3/ On QuesCom Gateway QWAdmin interface:

Change settings for GSM Device, SubscriberNumber should be the real E.164 number of the SIM.
Save config
Stop/start the QuesCom Gateway.