OZEKI NG SMS Gateway
"The World's most reliable SMS gateway software!"
info@ozekisms.com
Tel.: 00 36 52 532 731
Search the manual...
Overview | Quick start | Download | Manual | How to buy | FAQ | Contact Us

Automated phone calls?

Ozeki Phone System XE VoIP PBX software is an advanced PBX built for automated voice calls and 2 way SMS messaging. It has outstanding APIs for software developers. It can be used for:

Phone calls from SQL
Phone calls from HTTP
Voice and SMS applications

Callcenter developers

If you are working on telephone solutions, please check out the Ozeki VoIP SIP SDK.
It can be used to create VoIP client and server software.

Contact Us!

If you wish to get further information, do not hesitate to contact us!

E-mail: info@ozekisms.com

If you have a technical question, please submit a support request on-line.

Sending and receiving SMS From Java using the Ozeki Java SMS SDK

Share | G+1 | Tweet | Share

Download: ozeki-java-sms-sdkv4.zip (16 Kb)
Download: OzekiSmsClientv4.zip (10 Kb)

The Ozeki JAVA SMS SDK is your best choice if you want to send and receive SMS text messages to cellphones from a JAVA application. It was designed to be used in JAVA applications that have a GUI or that operate as a background service. It is a good choice if you want to send SMS messages, you need to react to incoming SMS messages or you would like to process SMS delivery reports instantly and in a resource efficient way. In case you need to add SMS functionality to a Java webservice, or to a JAVA solution where the JAVA process is executed for a very short time, this SDK is not your ideal choice. It is very efficient for long-term operations, but for short-lived processes other solutions might offer a better solution. For short-lived processes, we recommend you to use the JAVA HTTP SMS example or the JAVA SQL SMS example.

Introduction

Ozeki has released the Java SMS SDK, because it has experienced continuous demand from JAVA developers for a tool, that is able to add SMS functionality to JAVA applications in a very efficient way. This SDK communicates with the Ozeki NG SMS Gateway, through a TCP/IP socket. The socket is always connected, which makes it possible, to receive SMS delivery reports and incoming SMS messages instantly. The Ozeki Java SMS SDK implements the TCP/IP communication and provides methods calls and events you can implement to achieve the desired functionality. Using this SDK very fast and efficient SMS solutions can be developed. The SDK includes full source code. You are free to use and modify this code in your application.

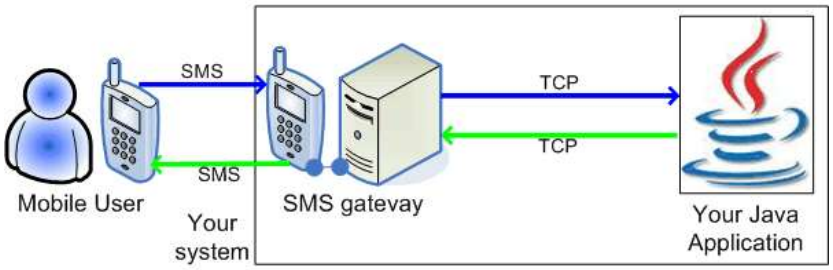

Figure 1 - Ozeki Java SMS SDK, communicates through a TCP/IP socket

To be able to use this SDK, you need to install Ozeki NG SMS Gateway into your corporate network. Ozeki NG SMS Gateway will be responsible for attaching your system to the mobile network. It will receive the TCP/IP connections from the JAVA SMS SDK and it well send and receive SMS messages through the configured communication method (Figure 1). For example you can send/receive SMS messages through a GSM phone attached to your computer with a phone to PC data cable, or you can send/receive SMS messages through the Internet if you have subscribed for an internet SMS Service.

Prerequisites

- Ozeki NG SMS Gateway
- Java SE Development Kit
- Eclipse IDE for Java developers
- Ozeki Java API

How to use the SDK

To use the SDK, you need to have the Java SE Development Kit installed on your computer. The SDK was developed and tested in Eclipse, but it should operate without a problem on any other JAVA development platform. The Ozeki java SMS SDK can be downloaded from the top of this page. It is a zip, file, that needs to be extracted into your project directory. Once extracted, run Eclipse IDE and add a new project.

The Ozeki SMS SDK contains a jar file called **ozekismsclient.jar**. You need to add it to the project by clicking on Project/Build Path/Add External Archives.

The next step to use it is to create a class which derives from the "OzSMSClient" ("MyOsSMSClient" in the following example).

```
public class MyOzSmsClient extends OzSmsClient
```

Create a constructor which will call "OzSmsClient"'s constructor.

```
public MyOzSmsClient(String host, int port) throws IOException,
    InterruptedException {
 super(host, port);
}
```

Bring the abstract methods of "OzSmsClient" into effect.

```
@Override
public void doOnMessageAcceptedForDelivery(OzSMSMessage sms) {
 Date now = new Date();
 System.out.println(now.toString() + " Message accepted for delivery. ID: " + sms.messageId);
}
@Override
public void doOnMessageDeliveredToHandset(OzSMSMessage sms) {
 Date now = new Date();
 System.out.println(now.toString() + " Message delivered to handset. ID: " + sms.messageId);
}
@Override
public void doOnMessageDeliveredToNetwork(OzSMSMessage sms) {
 Date now = new Date();
 System.out.println(now.toString() + " Message delivered to network. ID: " + sms.messageId);
```

```
}
@Override
public void doOnMessageDeliveryError(OzSMSMessage sms) {
 Date now = new Date();
 System.out.println(now.toString() + " Message could not be delivered. ID: " + sms.messageId +
  " Error message: " + sms.errorMessage + "\r\n");
}
@Override
public void doOnMessageReceived(OzSMSMessage sms) {
 Date now = new Date();
 System.out.println(now.toString() + " Message received. Sender
 address: " + sms.sender + " Message text: " + sms.messageData);
}
@Override
public void doOnClientConnectionError(int errorCode, String errorMessage) {
        Date now = new Date();
        System.out.println(now.toString() + " Message code: " + errorCode + ", Message: " + errorMessage);
}
```

Create an other class which we will utilize in practice.

```
public class TestTcpSms {
```

We set the availability of Ozeki NG.

```
String host = "localhost";
int port = 9500;
String username = "admin";
String password = "abc123";
```

Now start the connection with the Ozeki NG and log in.

```
MyOzSmsClient osc = new MyOzSmsClient(host, port);
osc.login(username, password);
```

Once you are logged in send a message.

```
if(osc.isLoggedIn()) osc.sendMessage("+001122334455", "Hello
     World");
```

Finally log off

```
osc.logout();
```

Example source code

Please feel free to edit and use this code in your project!

```
//MyOzSmsClient.java

package hu.ozekisms;

import java.io.*;
import java.util.*;
import hu.ozeki.*;

public class MyOzSmsClient extends OzSmsClient{

        public MyOzSmsClient(String host, int port) throws IOException, InterruptedException {
                super(host, port);
                // TODO Auto-generated constructor stub
        }


        @Override
        public void doOnMessageAcceptedForDelivery(OzSMSMessage sms) {
                Date now = new Date();
                System.out.println(now.toString() + " Message accepted for delivery. ID: " +
                sms.messageId);
        }

        @Override
        public void doOnMessageDeliveredToHandset(OzSMSMessage sms) {
                Date now = new Date();
                System.out.println(now.toString() + " Message delivered to handset. ID: " +
                sms.messageId);
        }

        @Override
        public void doOnMessageDeliveredToNetwork(OzSMSMessage sms) {
                Date now = new Date();
                System.out.println(now.toString() + " Message delivered to network. ID: " +
                sms.messageId);
        }

        @Override
        public void doOnMessageDeliveryError(OzSMSMessage sms) {
                Date now = new Date();
                System.out.println(now.toString() + " Message could not be delivered. ID: " +
                sms.messageId + " Error message: " + sms.errorMessage + "\r\n");
        }

        @Override
        public void doOnMessageReceived(OzSMSMessage sms) {
                Date now = new Date();
                System.out.println(now.toString() + " Message received. Sender address: " +
                sms.sender + " Message text: " + sms.messageData);
        }



        @Override
        public void doOnClientConnectionError(String errorCode, String errorMessage) {
                Date now = new Date();
                System.out.println(now.toString() + " Message code: " + errorCode +
                ", Message: " + errorMessage);
        }
```

```
}



//TestTcpSms.java

package hu.ozekisms;

import java.io.*;


public class TestTcpSms {

        /**
         * @param args
         */
        public static void main(String[] args) {
                try {
                        String host = "localhost";
                        int port = 9500;
                        String username = "admin";
                        String password = "abc123";

                        MyOzSmsClient osc = new MyOzSmsClient(host, port);
                        osc.login(username, password);


                        String line = "Hello World";

                        System.out.println("SMS message:");

                        if(osc.isLoggedIn()) {
                                osc.sendMessage("+551122334455", line);
                                osc.logout();
                        }


                } catch (IOException e) {
                        System.out.println(e.toString());
                        e.printStackTrace();
                } catch (InterruptedException e) {
                        System.out.println(e.toString());
                        e.printStackTrace();
                }
        }
}
```

Dig deeper!
People who read this also read...

- FAQ
- Java database SMS example
- Java HTTP SMS example
- Java SMS API
- Developers guide

Next page: VB.NET SMS API

Ozeki Cookie Policy
Ozeki Informatics Ltd uses cookies to provide you the best experience on this website. The further use of the website will be considered as an agreement to the use of cookies. For more information read this website.

Copyright © 2000 - 2016 Ozeki Informatics Ltd.
All rights reserved

Software PBX for Windows | VoIP SDK | Legal information | Privacy policy | Terms of use
Please, address your inquiries to info@ozekisms.com

POWERED BY OZEKI WEBOFFICE