 Copyright 2011 Click And Deploy (Pte. Ltd.)

AlertDispatcher v1.6 DLL API Guide

Last update: 1st July 2011

Table of Content

API Documentation

Overview

AlertDispatcherlib.dll allows developers to send messages (SMS/Email) and receive messages (new incoming SMS, outgoing SMS status updates) via events, and check AlertDispatcher server status from the network or even over the Internet.

AlertDispatcherlib.dll works on all Windows platform (from Windows 95 to Windows 7), and is not dependent on Dot Net or Java installations. A UNIX version of the component can be provided on request.

Properties

The following section shows all the properties that can be set for AlertDispatcherLib.dll.

Name	Type	Description
RemoteServerHost	String	IP address or Host Name where AlertDispatcher is running . Default: 127.0.0.1
RemoteServerPort	Integer	Port at which AlertDispatcher is listening. Default: 5556
ReceiveMessageUpdate	Boolean	Defines if the application is to be notified of message updates (both incoming and outgoing messages) via the MessageUpdate event. Default: True
ReadTimeout	Integer	Timeout for read operation in milliseconds. Default: 2000
APIParaphrase	String	Required if AlertDispatcher API is password protected.
ClientName	String	

Methods

The following section shows the methods that can be called for AlertDispatcherLib.dll.

SendSms

Send an SMS or Email to the specified recipients

```
SendSms(Recipients, Message, Subject, ScheduledTime, Priority, ModemPort, TrackingID)
```

Parameters:

Recipients	String	Comma separated list of recipients (phone number, email address or address books entries)
Message	String	Text of the message
Subject	String	Subject of the message (for the case of email message)
ScheduledTime	DateTime	Date and time to which the message will be scheduled for delivery
Priority	Integer	The priority of the message (1 - Lowest / 4 - Highest)
ModemPort	Integer	The modem port number to use for sending the message (0 = Auto)
TrackingID	String	[out] An automatic ID to track the message in events

Returns: *Operation Result ("ok" or error)*

CheckServer

Check the server status

```
CheckServer()
```

Returns: Operation Result ("ok" or error)

SetError

Allows API client to log an error on the Server. This error will be logged to AlertDispatcher event log, and automatically displayed on the AlertDispatcher Client.

```
SetError(Message);
```

Parameters:

Message	String	The error message text to send to the server

Events

The following section shows the events which you can register to receive notifications.

DebugLog

Raw socket log for debugging purpose

```
DebugLog(Message)
```

Parameters:

Message	String	The debug message

ConnectedServer

Event triggered when the connection to the Server is made

```
ConnectedServer()
```

DisconnectedServer

Event triggered when connected to and disconnected from Server

```
DisconnectedServer()
```

MessageUpdate

Receive Message record on status change (for both send/receive SMS/Email)

```
MessageUpdate(TrackingID, MessageStatus, InOut, PhoneNumber, Name, Message, Keyword,
ErrorText, DateTime, FinishDateTime)
```

Parameters:

TrackingID	String	ID of the message for tracking purpose. Its originally returned by the SendSms() method.
MessageStatus	Integer	Status of of the message Possible Values:

		0: Unknown 1: [Out]WaitingForSend 2: [Out]Sending 3: [Out]ReSending 4: [Out]Sent 5: Aborted 6: [Out]Error 7: [In]Received 8: [In]WaitingForSend 9: [In]Sending 10: [In]ReSending 11: [In]Sent 12: [In]Error 13: [Out]Waiting 14: [Out]PendingAcknowledge 15: [Out]Acknowledged 16: [Out]Escalated 17: [Out]AcknowledgeTimeout
InOut	Integer	Defines if the message is incoming or outgoing. Possible values: 1: In 2: Out
PhoneNumber	String	Phone number of the message
Name	String	Name of the entry in the address book
Message	String	Text of the message
Keyword	String	The first word of the message
ErrorText	String	Error description
DateTime	DateTime	Date-Time when the message was received by the server
FinishDateTime	DateTime	Date-Time when the message was actually sent

Register DLL manually

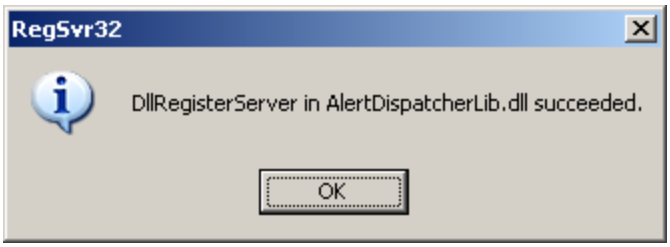
In order to use the DLL you must register it first. If you have installed the application via the setup, this is already taken care for you and you can skip this step.

If you want to register the DLL manually, you must first open a “Command prompt”,

From the command prompt, go into the folder where the DLL resides and type:

```
regsvr32 AlertDispatcherLib.dll
```

A screen similar to this one should appear:

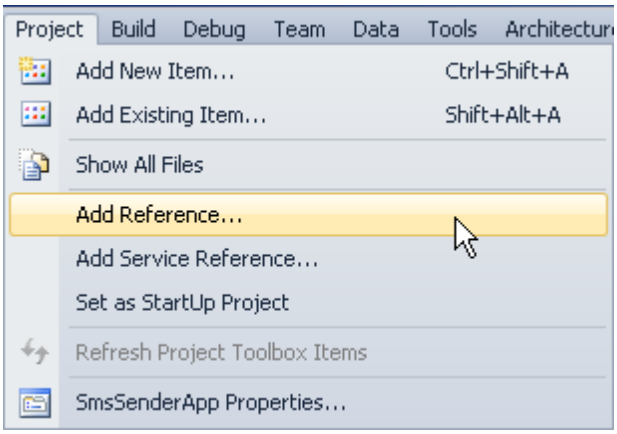


Note: In Windows Vista and Windows 7 you must open the “Command Prompt” with admin privileges (Run as Admin) in order to register the DLL.

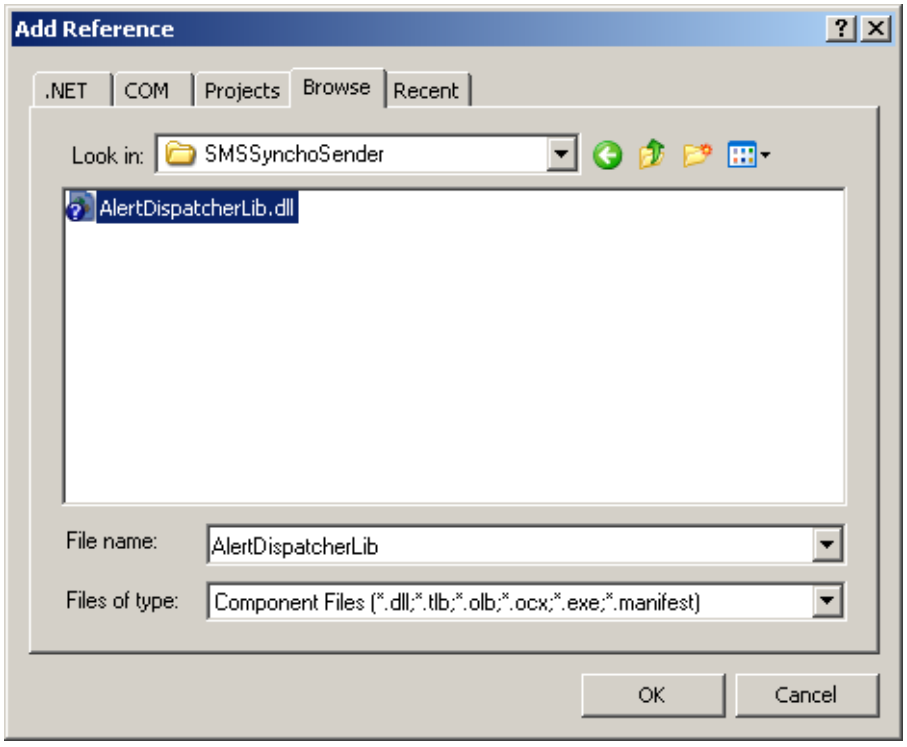
Working with Visual Studio

To be able to use the DLL in Visual Studio, you must first create a project or open the demo project which you can find under C:\Program Files\AlertDispatcher\APIs\DLL.

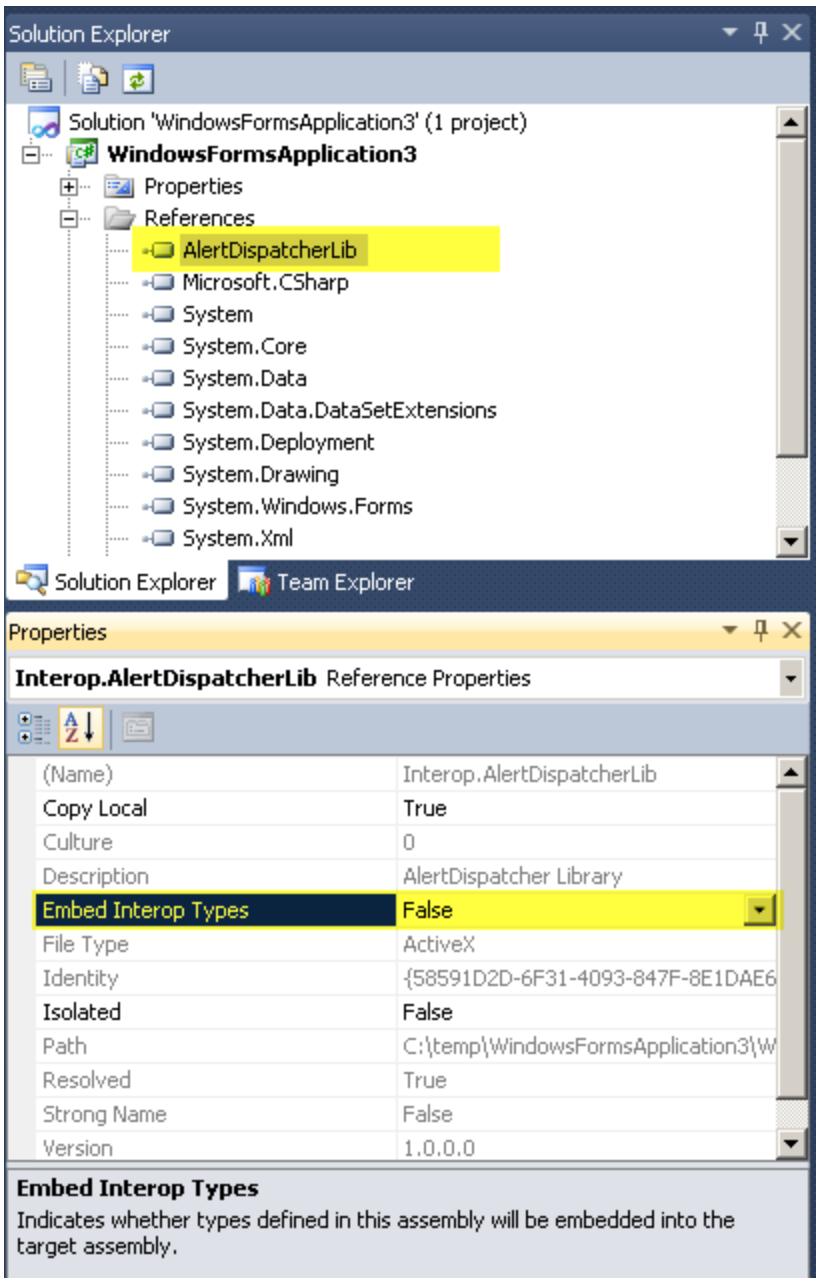
Then you must add the reference to the project, this is done from the “Add Reference...” menu as shown in the next screenshot:



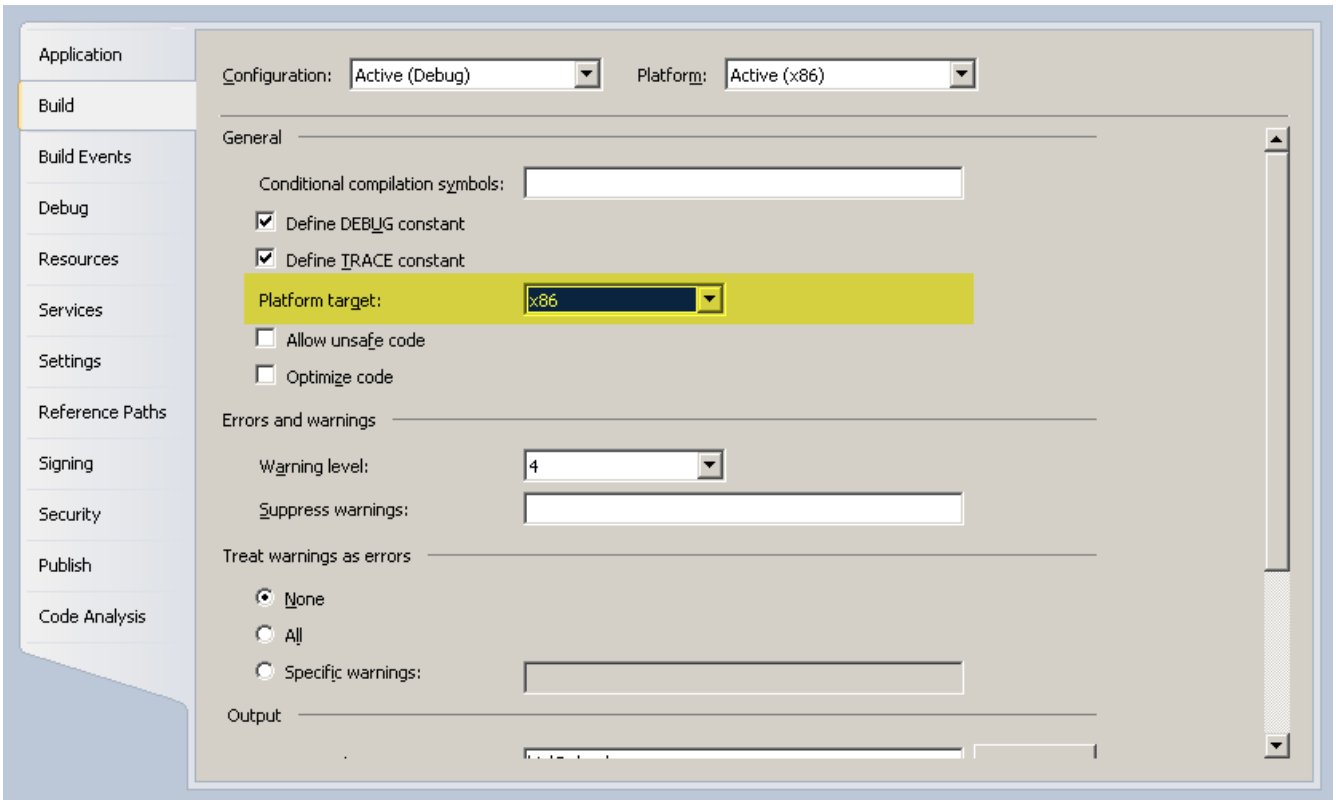
Next you must go into the “Browse” tab and find the DLL from your local drive. Press “OK” and the DLL is imported automatically. This will automatically generate the necessary interfaces and classes to communicate with AlertDispatcher Server.



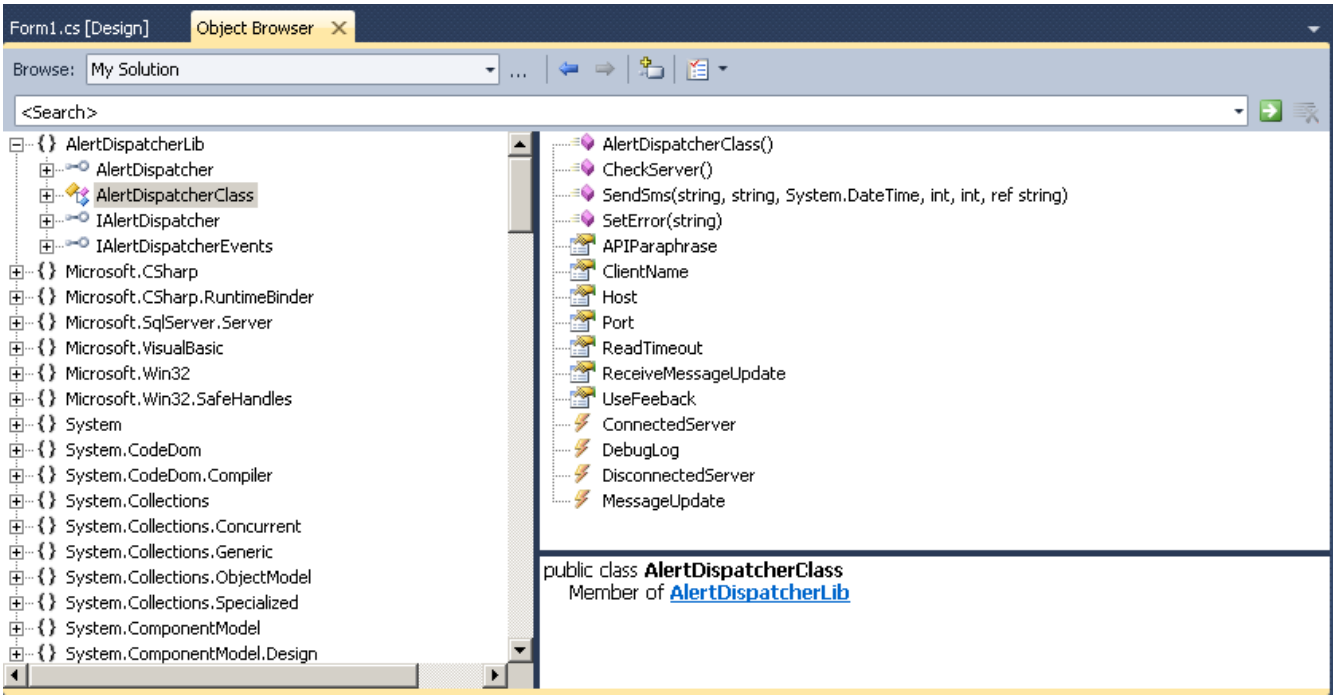
In case your project is targeted for .NET Framework 4, you must change the “Embed Interop Types” property to “False” in the AlertDispatcherLib reference property, as shown in the next screenshot, otherwise you cannot instantiate the AlertDispatcher class and hook the necessary events.



The AlertDispatcherLib.dll supports only 32 bit compilation for the moment, so if you are running in a 64 bit environment you must change the project properties to 32 bit as shown in the next screenshot:



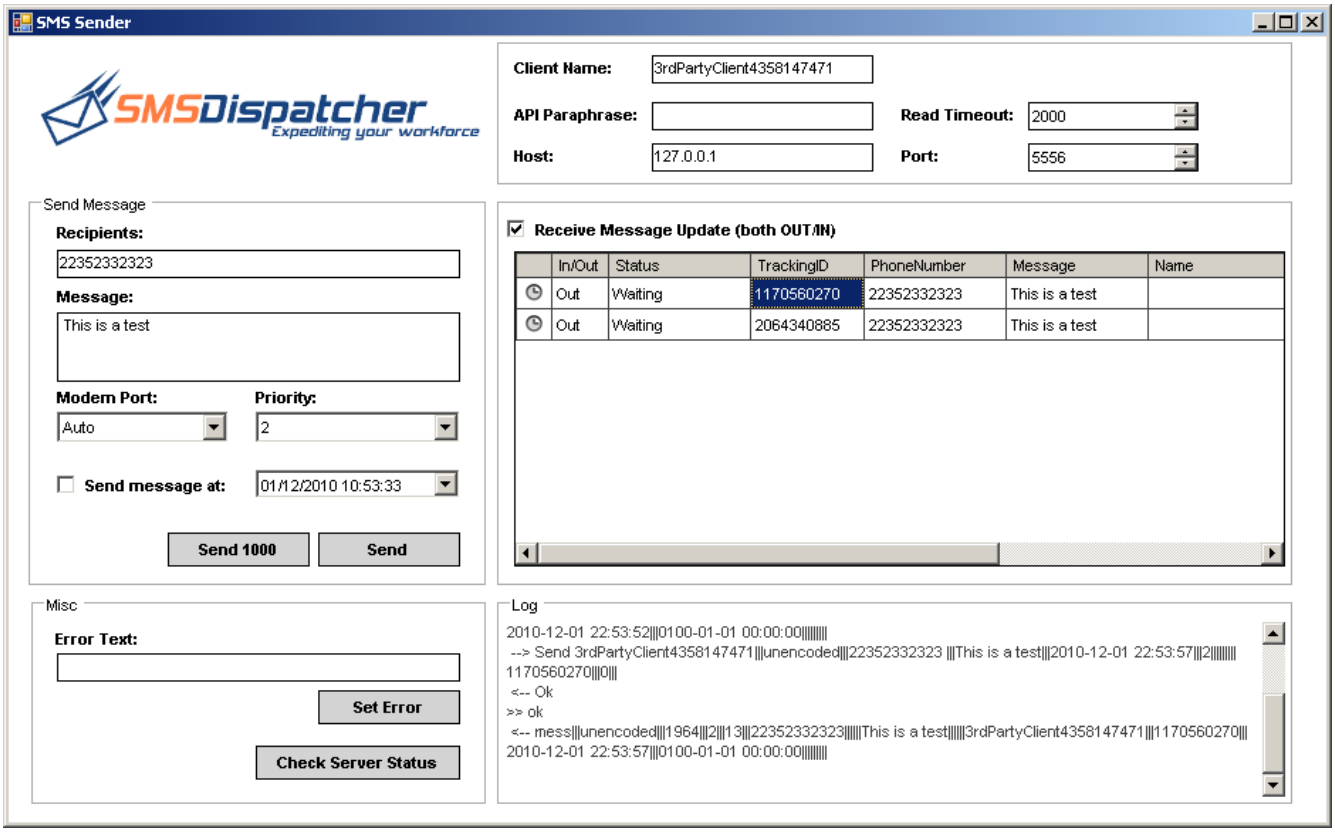
After you finish the import of the DLL, you can go to the into the Visual Studio Object Browser and take a look at the generated classes and methods, as shown in the next screenshot.



Code Sample Visual Studio

Sample Application

For complete code sample, please check the sample application provided under D:\Program Files\AlertDispatcher\APIs\DLL\C# Sample, which allows you to send SMS/Email, receive SMS and configure all the existing parameters. The sample application looks like this:



Code Snippets C#:

The following sections shows some code snippets with the initialization and the most common operations:

- Import the AlertDispatcherLib namespace:

```
using AlertDispatcherLib;
```

- Create the AlertDispatcher Class and configure the basic properties

```
AlertDispatcher dispatcher = new AlertDispatcher();
dispatcher.RemoteServerHost = "127.0.0.1";
dispatcher.RemoteServerPort = 5556;
```

- Register the events from the AlertDispatcher class, you must provide a function that will be called when the event ocurrs:

```
dispatcher.ConnectedServer += new
IAlertDispatcherEvents_ConnectedServerEventHandler(dispatcher_ConnectedServer);

dispatcher.DebugLog += new
IAlertDispatcherEvents_DebugLogEventHandler(dispatcher_DebugLog);

dispatcher.DisconnectedServer += new
IAlertDispatcherEvents_DisconnectedServerEventHandler(dispatcher_DisconnectedServer
);

dispatcher.MessageUpdate += new
IAlertDispatcherEvents_MessageUpdateEventHandler(dispatcher_MessageUpdate);
```

- Since the events are called within another thread context (a thread internal to AlertDispatcherLib.dll) you cannot directly change visual components data or an exception will raise, you must use some sort of Invoke as shown in the following example:

```
void logMessage(string message)
{

  if (memoLog.InvokeRequired) {
                memoLog.Invoke(new MethodInvoker(delegate {addMessage(message);}));
       } else {
          addMessage(message);
       }
}
```

- Send SMS:

```
string result = dispatcher.SendSms(tbTo.Text, tbMessage.Text, tbSubject.Text,
dateTime, cmbPriority.SelectedIndex + 1, cmbModemPort.SelectedIndex, ref
TrackingID);
```

Code Snippets Visual Basic:

- Create the AlertDispatcher Class and configure the basic properties

```
Dim dispatcher As New AlertDispatcher
dispatcher.RemoteServerHost = "127.0.0.1"
dispatcher.RemoteServerPort = 5556
```

- Register the events from the AlertDispatcher class, you must provide a function that will be called when the event ocurrs:

```
AddHandler dispatcher.DebugLog, AddressOf OnDebugLog
AddHandler dispatcher.ConnectedServer, AddressOf OnConnectedServer
AddHandler dispatcher.DisconnectedServer, AddressOf OnDisconnectedServer
AddHandler dispatcher.MessageUpdate, AddressOf OnMessageUpdate
```

- Send SMS:

```
Dim Result = dispatcher.SendSms(PhoneNumber, Text, Subject, Date.Now, 2, 0,
TrackingId)
```

Working with Java

Java 64 bit

The AlertDispatcherLib.dll supports only 32 bit compilation for the moment, so if you are running in a 64 bit environment you must download the JRE 32 bit version or you will get an error like "*Can't load IA 32-bit .dll on a AMD 64-bit platform*".

You can download JRE 32 bit version from here:
http://www.oracle.com/technetwork/java/javase/downloads/index.html

Select "Download JRE" and on the next screen select "Windows" as operating system (not Windows x64).
When you execute the provided samples, make sure you are running the correct JRE 32 bit version, which is usually installed under "`C:\Program Files (x86)\Java\jre6\bin`"

com4j

To be able to use the DLL in Java, you must need some external library to do the bind between the java code and the DLL.

In this sample we use com4j (https://com4j.dev.java.net/) but another alternatives are possible.

Once you have registered the DLL (see *Register DLL manually*) and download the com4j, you can proceed to use the com4j tool for generating the Java classes to use the library.

```
java -jar tlbimp.jar -o . -p alertdispatcher AlertDispatcherLib.dll
```

This will generate a alertdispatcher package in the current directory.

You can avoid this step and directly use the *alertdispatcher* package provided in the sample (*alertdispatcher.jar)*

You must include the *com4j.jar* and *alertdispatcher.jar* within your application to be able to compile and run.

Usage

- Create the dispatcher class and set the basic properties. (Note: Property names are case sensitive.)

```
dispatcher = ClassFactory.createAlertDispatcher();

dispatcher.useFeedback(true);
dispatcher.readTimeout(2000);
dispatcher.receiveMessageUpdate(true);
dispatcher.remoteServerHost("127.0.0.1");
dispatcher.remoteServerPort(5556);
```

- Define a class to handle the events, in this sample it only prints in the console the event name or some debugging info, the class must extents “IAlertDispatcherEvents” class

```
public class AlertDispatcherEvents extends IAlertDispatcherEvents {

    @DISPID(201)
    @Override
    public void debugLog(java.lang.String message) {
        System.out.println(message);
    }

    @DISPID(202)
    @Override
    public void connectedServer() {
        System.out.println("<< Connected");
    }

    @DISPID(203)
    @Override
    public void disconnectedServer() {
        System.out.println("<< Disconnected");
    }

    @DISPID(204)
    @Override
    public void messageUpdate(
            java.lang.String trackingID,
            int messageStatus,
            int inOut,
            java.lang.String phoneNumber,
```

```
            java.lang.String name,
            java.lang.String message,
            java.lang.String keyword,
            java.lang.String errorText,
            java.util.Date dateTime,
            java.util.Date finishDateTime) {
        System.out.println("<< New Message Status");
    }
}
```

- Hook the events, associate the dispatcher with the sample event handler class

```
EventCookie cookie = dispatcher.advise(IAlertDispatcherEvents.class, new
AlertDispatcherEvents());
```

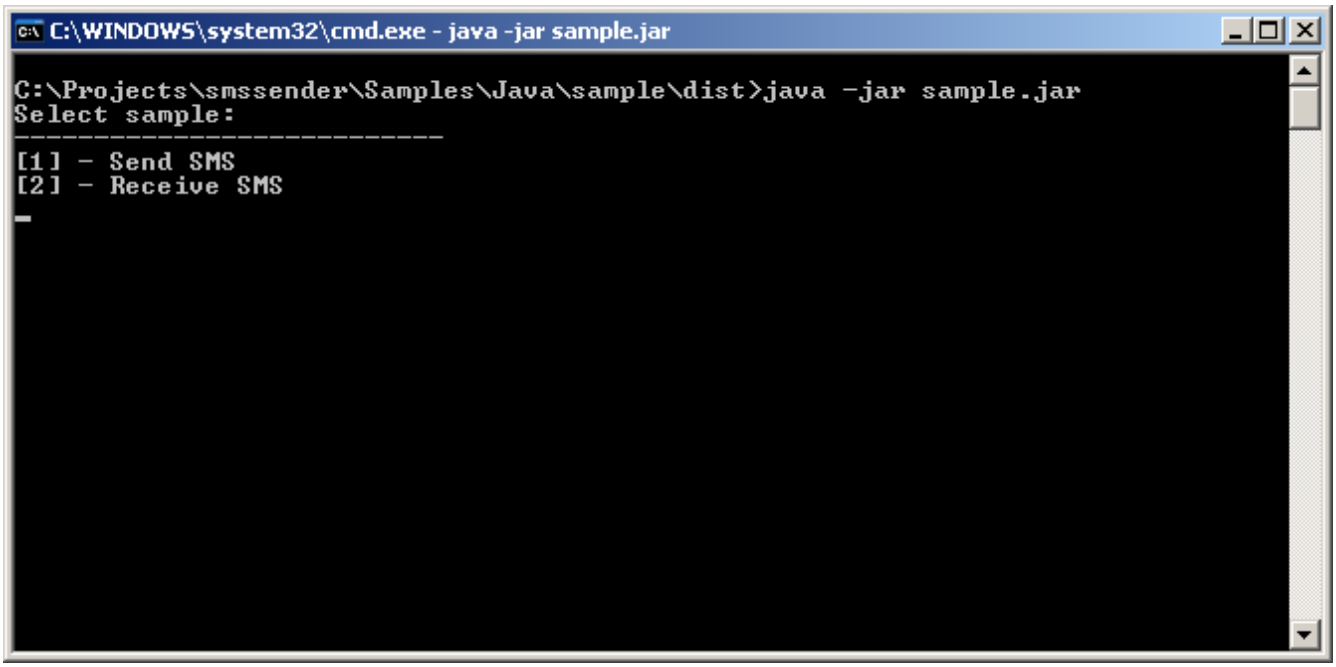
- Send a SMS

```
Holder<String> trackingID = new Holder<String>();
String result = dispatcher.sendSms(phoneNumber, text, subject, new
Date(), 2, 0, trackingID);
```

- Testing the sample application: Go into the command line, cd into the *"dist\"* folder inside *"sample\"* and type:

```
java -jar sample.jar
```

You will see a screen like the following where you can test the basic functionality of the library:


C:\WINDOWS\system32\cmd.exe - java -jar sample.jar
C:\Projects\smssender\Samples\Java\sample\dist>java -jar sample.jar
Select sample:
[1] - Send SMS
[2] - Receive SMS