skype journal
independent news, views, and support

SKYPE API GUIDE:

Learning Skype's Plug-In Architecture

FIRST EDITION. APRIL, 2005

DEMONSTRATION PROJECT: BUILD YOUR OWN SKYPE ANSWERING MACHINE.

- UNDERSTAND & WORK WITH THE SKYPE API
- BUILD TOMORROW'S COMMUNICATION PLATFORM
- CUSTOMIZE SKYPE SOLUTIONS
- CREATE POWERFUL APPLICATIONS, STEP-BY-STEP

"I think this is a very good book for the novice programmer and more importantly it is an excellent book for an experienced programmer like me (26 years!), but who is new to Skype.

Thank-you, Bill."

Arie Bakker, Netherlands

By W.J. (Bill) Campbell,
Technical Editor, Skype Journal and the Skype Developer Community

Available in: Chinese, Dutch, English, German, Hungarian, Italian, Japanese, Polish, Portuguese, Romanian, Russian and Spanish

USEFUL LEGAL STUFF ON SKYPE

Helpful links to understand the Skype End User License Agreement (EULA) and associated issues.

Search the knowledge base:
http://support.skype.com/?_a=knowledgebase&_j=subcat&_i=13

Certification:
www.skype.com/partners/certification/

Terms of use:
www.skype.com/company/legal/

End User License Agreement:
www.skype.com/company/legal/eula/index.html

Distribution Terms:
www.skype.com/company/legal/promote/distributionterms.html

Promotional terms:
www.skype.com/company/legal/promote/materialterms.html

Banners:
www.skype.com/community/promoteskype.html

Acknowledgements

I would like to acknowledge the huge contributions that many other fellow Skype enthusiasts provided to this book. It simply would not exist without their kind and generous support. The enthusiasm of these contributors demonstrates the excitement that the Skype API is generating internationally.

CONTRIBUTORS (see page 56):

- Arie Bakker
- Kevin Delaney
- German Koninin
- Johnny Krogsgaard
- Wei Li
- Gaurav Prasad
- Jason Terando
- beeSync

TRANSLATORS (see page 58):

- Ewelina Chudzińska
- Markus Daehne
- Ramon
- Brian Harward
- Peter Henning
- Seishi Isobe
- kootstra
- Nina Kupper
- Tracizio Pinto
- Paulo Sousa
- Wuyijun
- Skaipe.com

TESTERS (see page 61):

- Cesar Andrade
- ascenna
- Arie Baker
- Vir Banhu
- Peter Henning
- jaragrets
- Neil Lindsey
- Jerry Pedersen
- sanith1976
- Rolf Stefanni
- Tomasz Tybulewicz
- Yann

TABLE OF CONTENTS

Acknowledgements 3

Learning Skype's Plug-In Architecture 5

What is Skype API COM Wrapper & SkypeX? 6

The Skype API Documentation 7

Understand & Work with the Skype API 7

Install the Skype API COM Wrapper 10

Create Powerful Applications 15

Install SkypeX 21

APPENDIX 26

- Visual BASIC.NET Sample Chat Message Utility: 26
- Perl Sample Chat Message Utility: 28
- PHP GTK Sample Chat Message Utility: 36
- Java Sample Chat Message Utility: 40
- Programming Guide for Microsoft C# 46
- Programming Guide for Microsoft C++ 49
- Using the CSkypeMessageQueue Class 55
- Contributors: 56
- Translators: 58
- Testers: 61
- About the Author: 62

Learning Skype's Plug-In Architecture

INTRODUCTION

This guide introduces the Skype Application Programming Interface (API) to both experienced programmers who are new to Skype and to novice programmers. Maybe you're not a programmer, but you have a common question, "*Can this be done with the Skype API*?" Or maybe you just have a natural curiosity to understand how things work. If any of these is the case, then this guide is for you.

Its approach is practical and simple. Read on. Together we'll explore the API. Within five minutes you'll be sending your first Chat Message via the API using a Skype third-party utility. Then you'll install the API Component Object Model (COM) wrapper that allows programs to be written in a number of different programming languages to access the Skype API. We'll even show you how to create a simple Skype Answering Machine.

Many novice programmers only understand one programming language. They find it difficult to get started with the API because they can't read the C++ or Virtual BASIC 6 examples that are commonly available. So we've tried to include a simple example for each language. This version of the API Guide won't have them all, but this is a living document and it will be updated frequently.

Skype is building what looks like a new and incredibly large market. The release of the Skype API allows third-parties to participate in this opportunity for learning, fun or profit.

One final note: this guide is a step-by-step guide. It works like any computer program you write. Skip a step, or start somewhere other than the beginning and you will get an error. ☺

The Skype API Guide at a Glance

1. Download Skype API Documentation
2. Download SkypeTracer utility
3. Send Chat Message
4. Make PSTN Call
5. Download and install Skype API COM Wrapper
6. Download Chat Message Utility
7. Send Chat Message
8. Review commented code for Chat message utility

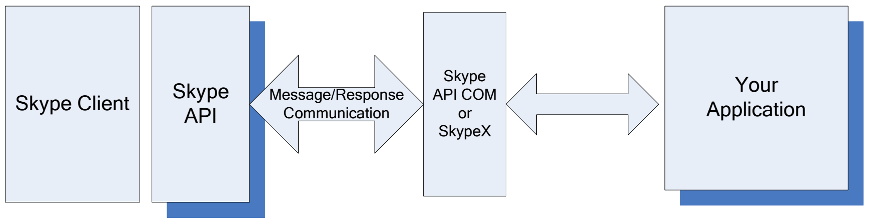
What is Skype API COM Wrapper & SkypeX?

First, what is the Skype API?

The Skype Application Programming Interface (API) allows external programs to access Skype functionality. The Skype API, which uses a “message-response” architecture, requires those external programs be written in C++. That would leave a lot of us out of the game if it were not for (free) products like *Skype API COM Wrapper* and *SkypeX*. These products allow external programs to be written in other programming languages, such as Visual BASIC (VB) 6, Visual BASIC.NET, PHP, Java and Perl.

Skype API COM Wrapper and SkypeX wrap around the Skype API Windows Messaging Environment. COM wrappers (referred to as “wrappers”, “glue” or “middleware”), using Microsoft’s Component Object Model (COM), provide an elegant way to interface products.

A COM wrapper is an object-oriented set of components that programmers use to design an interface that is efficient and easy to maintain. In this case, it provides for easy, rapid integration of Skype into MS Office products, in particular SharePoint and Internet Explorer, as well as other software applications.



The Skype API COM Wrapper and SkypeX provide another unique and important function for developers planning to interface a hardware device or software application to the Skype API: these wrappers enable rapid prototype development.

Let’s get started…

The Skype API Documentation

First, download the Skype API Documentation (*Skype API – Description of Skype API and how to use it*) that describes the message-response commands used to access Skype functions, from here.

Second, read "Expanding Skype" on Skype Community website, here.

Tip: To access the standard Skype API you need to be highly skilled in C++. Luckily a number of developers have created an API COM Wrapper that provides access to the Skype API via a number of simpler programming languages like:

- Visual BASIC 6
- VB.NET
- PHP
- Perl
- Java
- C#

Understand & Work with the Skype API

Before we get any deeper into technical stuff let's use the API to send a Chat Message.

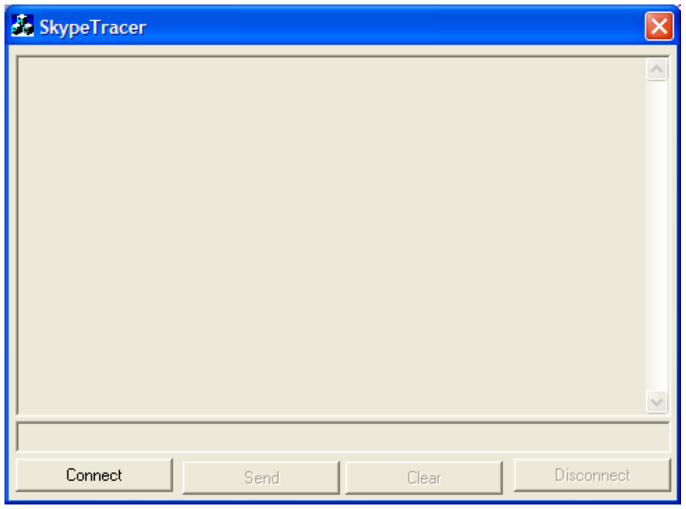
Start here: download and install this little tool – SkypeTracer.

(Reference: http://forum.skype.com/viewtopic.php?t=18150)

SkypeTracer is a neat little utility to monitor all the activity in your Skype client and issue commands via the Skype API. SkypeTracer displays a real-time listing of activity within Skype and communication to and from the outside world.

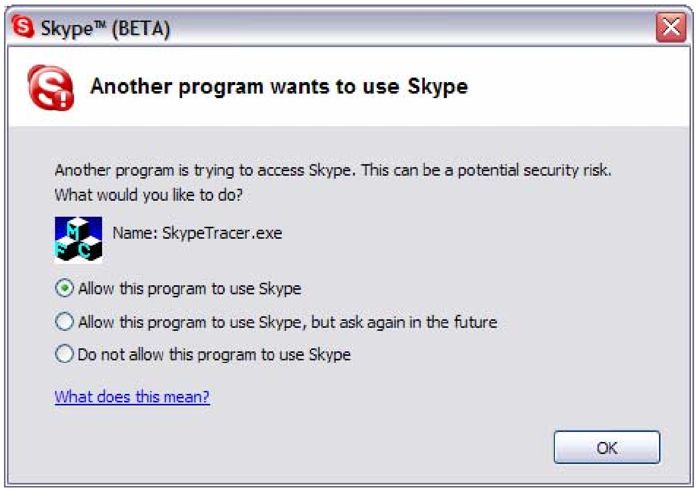
As "iunknown" the author of the SkypeTracer says, "*Put SkypeTracer in your left hand and the API Reference Manual in your right and you are on your way to using the API*".

SkypeTracer's opening screen:



To start SkypeTracer, click "Connect".

You'll see a notice window like the one below. Select "Allow this program to use Skype". Click OK.



Skype™ (BETA)

Another program wants to use Skype

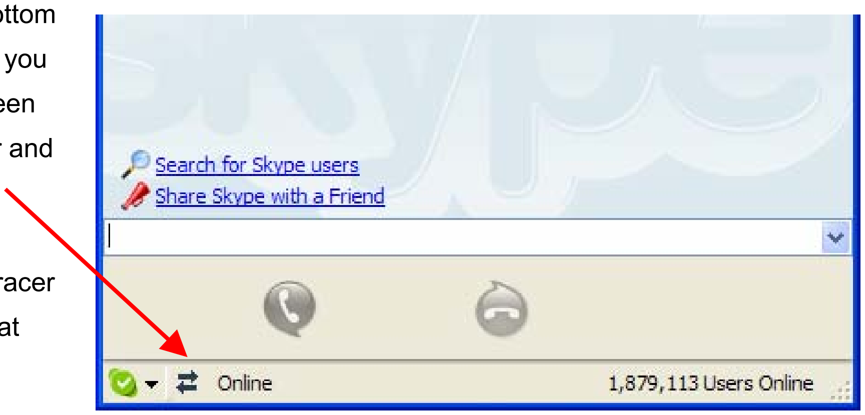
Another program is trying to access Skype. This can be a potential security risk.
What would you like to do?

Name: SkypeTracer.exe

[✓ Allow this program to use Skype]
Allow this program to use Skype, but ask again in the future
Do not allow this program to use Skype

What does this mean?

Note the double arrows at the bottom left of your Skype window telling you that a communication link has been set up between the SkypeTracer and your Skype client.



Now we're ready to use SkypeTracer and the Skype API to send a Chat Message!

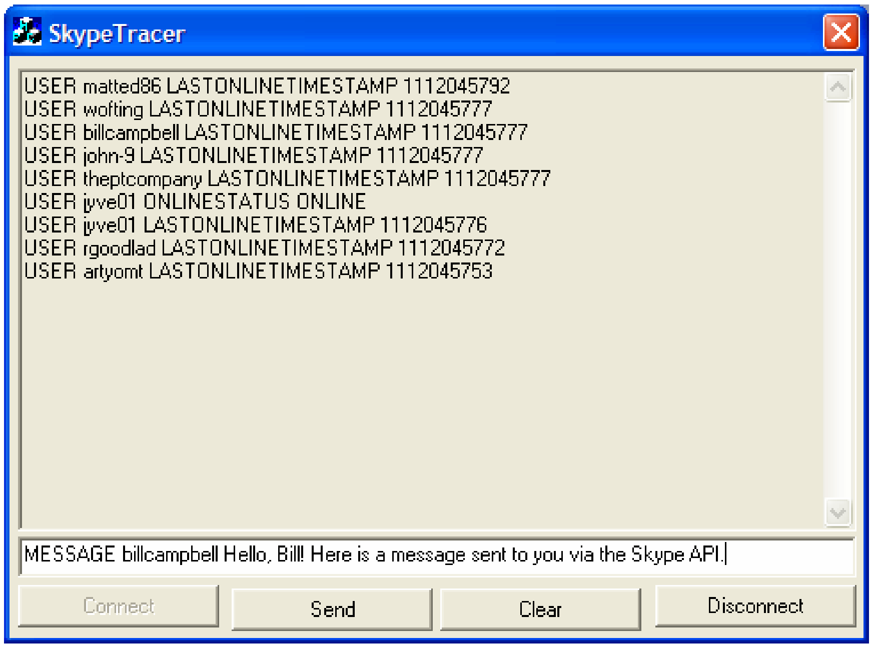
Open or print the Skype API Documentation and turn to section 7.9.5, “Sending Messages”.

The Skype API Reference Manual gives us the format and syntax for a command to send a chat message to any Skype User who’s Skype Name we know:
`MESSAGE UserName Text`
Where “UserName” is replaced with the Skype Name and “Text” is replaced with the text of your message.

Send a Chat Message to one of my Skype Clients with the Skype Name of `billcampbell`

We enter the command exactly like this into the command line:
```
MESSAGE billcampbell
Hello, Bill. Here is a
message sent to you
via the Skype API.
```

Click “Send” to execute.



```
USER matted86 LASTONLINETIMESTAMP 1112045792
USER wofting LASTONLINETIMESTAMP 1112045777
USER billcampbell LASTONLINETIMESTAMP 1112045777
USER john-9 LASTONLINETIMESTAMP 1112045777
USER theptcompany LASTONLINETIMESTAMP 1112045777
USER jyve01 ONLINESTATUS ONLINE
USER jyve01 LASTONLINETIMESTAMP 1112045776
USER rgoodlad LASTONLINETIMESTAMP 1112045772
USER artyomt LASTONLINETIMESTAMP 1112045753
```

```
MESSAGE billcampbell Hello, Bill! Here is a message sent to you via the Skype API.
```

The browser window displays a real-time listing of activity in Skype and communication to and from the outside world.

You have just demonstrated your Skype API expertise!

The Status Window in SkypeTracer will list all the communication regarding your sent message. Good luck finding it amongst all the other online status changes being communicated, but it is there. ☺

You can test most of the commands listed in this Skype API Guide in this same way to get an understanding of the functions available to you. For example, if you have a SkypeOut account, type in CALL followed by +12505551212…. Bingo! You have initiated a SkypeOut call!

(Refer to the Skype API Documentation, section 7.9.4, “Making Calls”.)

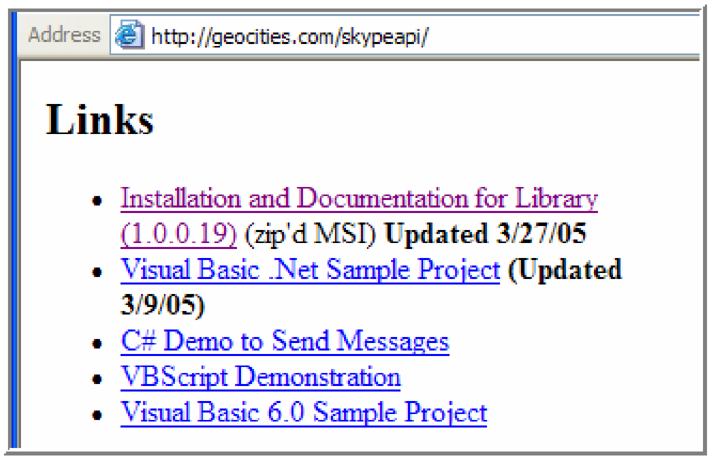
Install the Skype API COM Wrapper

Now that you've mastered the use of manually using the API, let's write a Visual BASIC program to do this automatically.

Close the SkypeTracer utility.

Next, download and install the Skype API COM Wrapper that will allow you to communicate with the API using Visual BASIC instead of C++, from here.

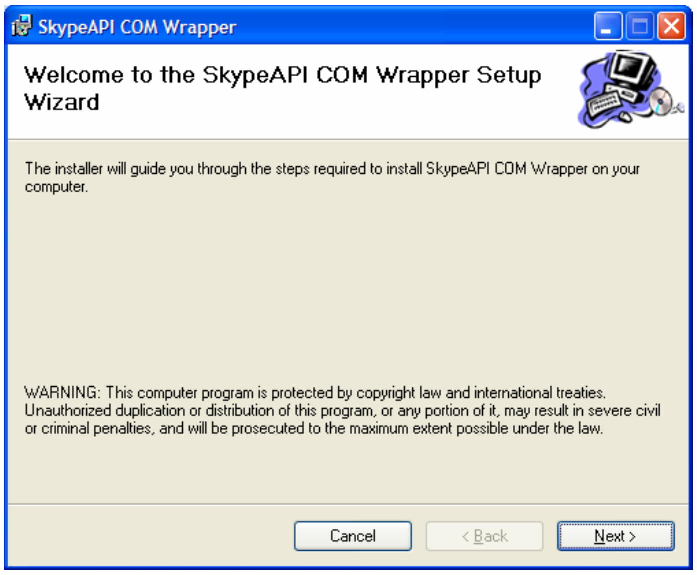
This Skype API COM Wrapper was written by Jason Terando, Huntington Beach, California, USA. His Skype Forum Handle is `caraccas`.



Links

- Installation and Documentation for Library (1.0.0.19) (zip'd MSI) **Updated 3/27/05**
- Visual Basic .Net Sample Project **(Updated 3/9/05)**
- C# Demo to Send Messages
- VBScript Demonstration
- Visual Basic 6.0 Sample Project

Double click on the SkypeAPIInstall icon to install.



Welcome to the SkypeAPI COM Wrapper Setup Wizard

The installer will guide you through the steps required to install SkypeAPI COM Wrapper on your computer.

WARNING: This computer program is protected by copyright law and international treaties. Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.

The Skype API COM Wrapper is now installed and registered.

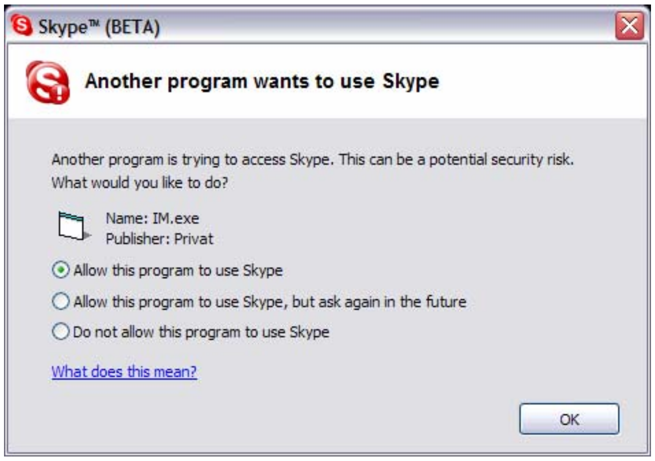
SAMPLE PROGRAMS FOR THE SKYPE API COM WRAPPER

Now that the Skype API COM Wrapper is installed and registered let's test it.

Download the "chat.zip" file from here.

Extract the file to your Desktop. Double click on IM.exe.

You'll see a notice window like the one below. Select "Allow this program to use Skype".


Skype™ (BETA)

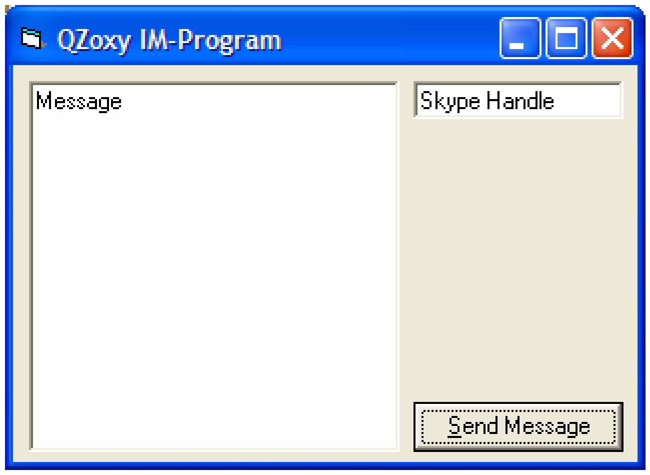
Another program wants to use Skype

Another program is trying to access Skype. This can be a potential security risk.
What would you like to do?

Name: IM.exe
Publisher: Privat

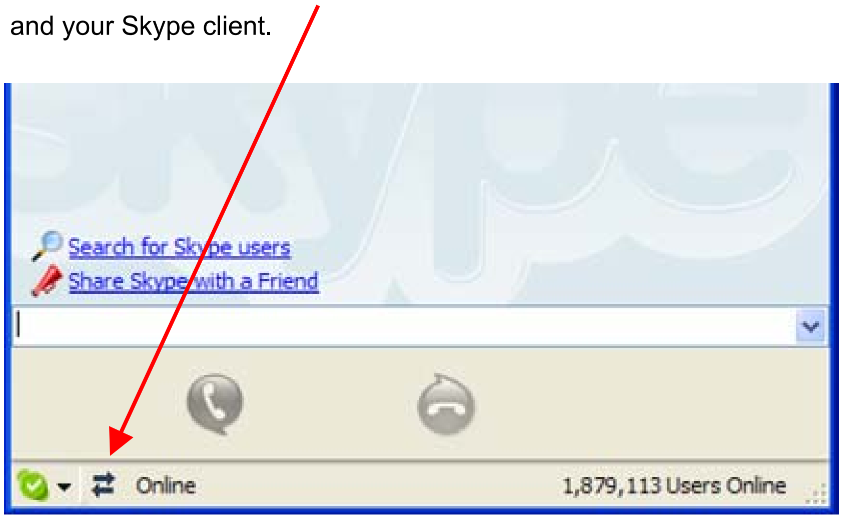
[✓ Allow this program to use Skype]
Allow this program to use Skype, but ask again in the future
Do not allow this program to use Skype

Type your recipient's Skype Name in the field that says "Skype Handle" and your text message in the "Message" box.



Click "Send Message"

Note the double arrows at the bottom left of your Skype window telling you that a communication link has been set up between the IM Program and your Skype client.



Below you'll find the commented source code for the sample IM.exe program.

COMMENTED SOURCE CODE FOR THE SAMPLE CHAT MESSAGE PROGRAM

```
Option Explicit

Private m_objConversion As SKYPEAPILib.Conversion
```

'Declare Variable of type SKYPEAPILib.Conversion

```
Private WithEvents m_objSkype As SKYPEAPILib.Access
```

'Declare Objects m_objskype as SKYPEAPILib.Access

'also look

```
Private Sub Form_Load()
```

'This function is automatically fired when form is getting started or loaded for first time so good place to initialize Objects declared earlier

```
Set m_objConversion = New SKYPEAPILib.Conversion

    On Error GoTo NoInit ' error handler,
    Set m_objSkype = New SKYPEAPILib.Access
```

'Initialize   m_objSkype

```
    m_objSkype.Connect
```

‘Connect to Skype network

```
    Exit Sub

NoInit:
    MsgBox Err.Description, vbCritical + vbOKOnly,
"Unable to Connect to Skype"
```

‘If unable to connect to Skype network for any runtime issues like network issues, etc., show error description to user and unload form

```
    Unload Me
    Exit Sub
End Sub

Private Sub btnSendCommand_Click()
```

‘This will be fired when user clicks on send button

```
Dim s As String

    On Error GoTo NoCommand
```

'error handler

```
    s = "MESSAGE " & handle.Text & " " & message.Text
```

‘This just appending text you have written in text box to string s declared above and constructing a command string to be passed to SKYPEAPILib.Access instance

```
 If Len(s) = 0 Then
```

‘Check if length of message is zero if yes then exit

```
    Exit Sub
    End If
    m_objSkype.SendCommand s
```

'Pass the command string to SKYPEAPILib.Access which tells it to send the message (message.Text)

```
  Exit Sub

NoCommand:
    MsgBox Err.Description, vbCritical + vbOKOnly,
"Unable to Send Command"
```

‘If any runtime error occurs show error

```
End Sub
```

You'll find the code for this simple Chat Message application in other programming languages, such as PHP, in the appendix.

Tip: Monitor the API issues by visiting the Skype Forum. For example if you look at this thread it will get you started using PHP.

GIVE YOUR PROGRAM AN INTERNATIONAL SCOPE

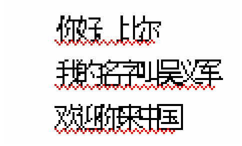
Skype is international and therefore uses Double Byte coding to handle international character sets such as those found in the Asian Character Set. Skype uses UTF8 everywhere throughout its interface and in the messaging function. So should you.

If you're a C/C++ programmer, use WCHAR arrays or the wstring standard library class to store strings in your application. If you're communicating directly with Skype, you can use the MultiByteToWideChar to set up your WM_COPYDATA messages. Conversely, when receiving a message from Skype, you can use the MultiByteToWideChar function to translate the incoming UTF8 to Unicode. Visual BASIC programmers can also use these methods but will need to import the function prototypes using the API Text Viewer.

.NET programmers can use the Encoding.UTF8 encoding class's GetBytes and GetString methods to translate between byte arrays and strings.

Copy and paste the test message from the text box below to test out your own Chat Messaging programs for UTF8 compatibility: If you see a series of boxes instead of the Chinese Characters shown below right then you need to enable Asian Language support in Control Panel>Regional & Language.

你好，比尔
我的名字叫吴义军
欢迎你来中国



你好，比尔
我的名字叫吴义军
欢迎你来中国

Translation:
hello, Bill
my name is wuyijun
welcome to China

Create Powerful Applications

In this chapter you will take what you have covered so far, add a few more Skype API commands to your repertoire and then we will guide you to create a simple Skype Answering Machine Application.

You will need the copy of the Skype API Documentation printed or online for this section of the guide. You probably already have downloaded this as per instructions on page 7. ☺ But if you need to now, the link is here.

We're not going to rewrite the Skype API Documentation, but we would like to discuss three simple commands so non-technical users will have an easier time accessing the Skype API Documentation.
The three commands types we want to discuss are:

- Query
- Set
- Execute

Query allows you to make an inquiry about status of the local Skype Client. For example, if you want to know what Audio Input Device is currently used by the Skype Client, you simply use the command (ALL CAPS):

```
GET AUDIO_IN
```

You can test this using the SkypeTracer Utility introduced on page 7.

Set parameters in the local Skype Client. This allows you to change a status of the Skype Client. For example, suppose you want the input audio in the Skype Client to routed to your Logitech 4000 Pro Video microphone instead of your Plantronics headset:

```
SET AUDIO_IN Logitech Microphone (Pro 4000)
```

Execute a function in the local Skype Client. You already did this back on page 9. Let's do it again:

```
MESSAGE billcampbell Hello Bill this is a test
message!
```

These three example commands can be the framework for you to develop a simple Skype answering machine. Let's see how.

Our simple Skype Answering Machine will have the following functions:

1. Answer a call
2. Play a recorded voice greeting
3. Allow the caller to record a message.

Specifications: our simple Skype Answering Machine will—

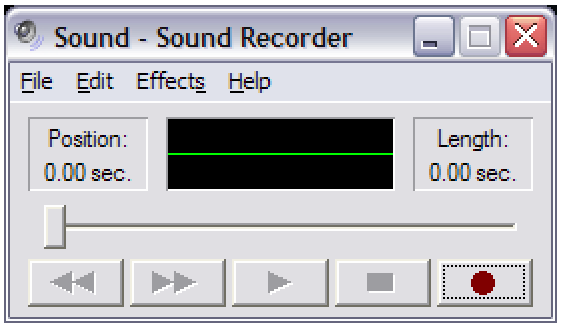
- Run on XP Operating System
- Be developed in VB.NET
- Use Windows Sound Recorder for playback and recording
- Use MS Multimedia Control to manage Windows Sound Recorder from here.
- Use the single (free) Virtual Audio Cable. Download this virtual audio driver from here.

Tip: How do you handle the Skype sound channel?
The Skype API does not in itself expose the sound channel via the Skype API. However it does allow you to select what audio devices connect to the Skype Client. In this way you can route audio in and out of your Skype Client. This gives you the ability to create applications such as Answering Machines, Call Recording (Skype PodCaster), and Dual Tone Multi-Frequency (DTMF) driven Interactive Voice Response (IVR) Systems.

Let's get started on creating a very simple Skype Answering Machine:

I use a Plantronics headset so these examples show that in the Default devices selected. Simply replace "Plantronics headset" with your system's speakers and microphone available from the dropdown menus.

Use Windows' *Sound Recorder* to create a "Greeting Message". (Sound Recorder is found at Start>All Programs>Accessories>Entertainment> Sound Recorder. Create a short cut for this application on your Desktop)



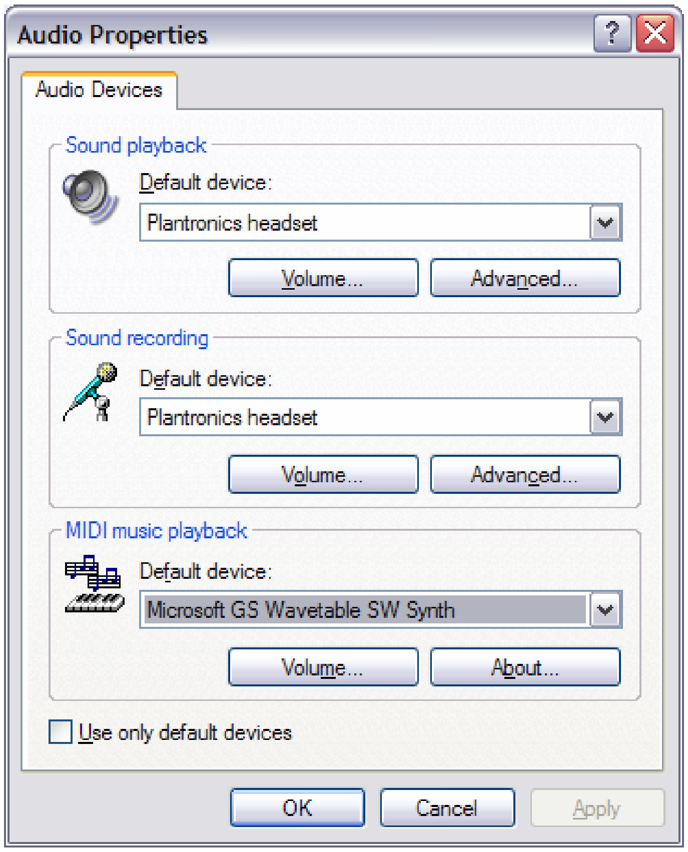
Set the properties for Sound Recorder to record from your microphone (in my case, the Plantronics headset). Properties are found at Edit>Audio Properties>Audio Devices tab of the Sound Recorder.

Then record your personal "Greeting Message". Save the file with the file name: Greeting Message.wav in a Folder with an appropriate name like "My Skype Answering Machine".

Now let's set up your Skype client up so you can play this Greeting Message when a when a contact calls you.

To set up your answering machine, you'll need the Virtual Audio Cable. Download and install the free Virtual Audio Cable (if you haven't already done so) from here. Once installed, reboot your computer.

Think of Virtual Audio Cable as a wire cable that you use to hook up any device to your sound card or stereo equipment.


Audio Properties

Audio Devices

Sound playback

Default device: Plantronics headset

Sound recording

Default device: Plantronics headset

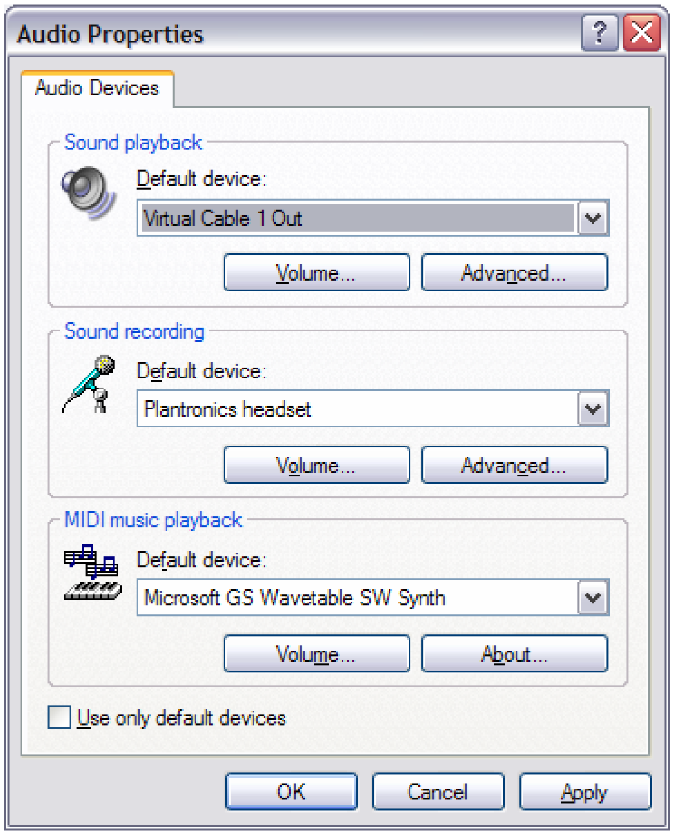
MIDI music playback

Default device: Microsoft GS Wavetable SW Synth

Initially, what you'll want to do is hook the audio output (speaker) from the Sound Recorder to the audio input (microphone) of your Skype client.

This "virtual" connection is made by editing the Audio Devices in Sound Recorder and the Sound Devices in Skype.

Audio Devices in Sound Recorder (Sound Recorder>Edit>Audio Properties>Audio Devices tab):
For Sound playback, select "Virtual Cable 1 Out" as the Default device.


Audio Properties

Audio Devices

Sound playback

Default device: Virtual Cable 1 Out

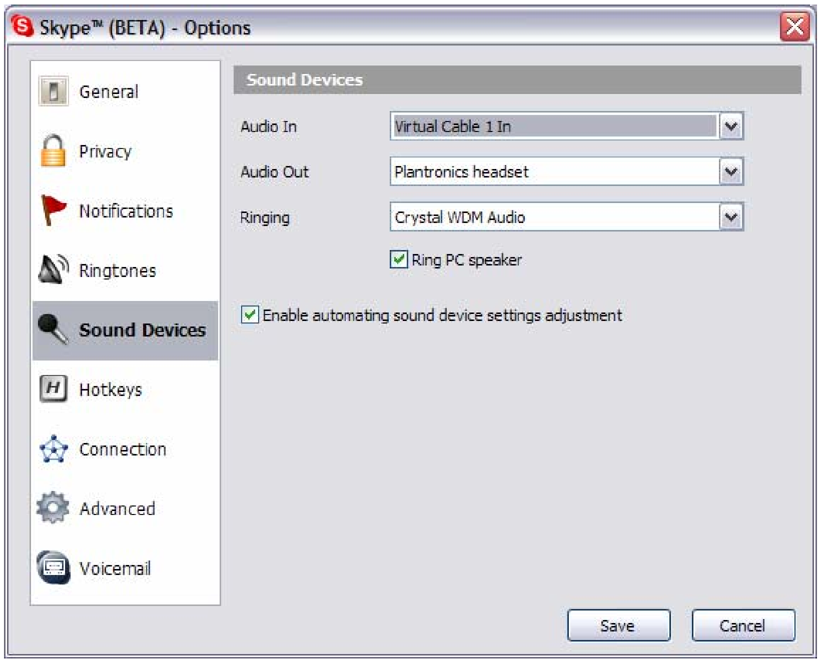
Sound recording

Default device: Plantronics headset

MIDI music playback

Default device: Microsoft GS Wavetable SW Synth

Sound Devices in Skype (Skype> Tools>Options>Sound Devices): For Audio In, select “Virtual Cable 1 In”.



Sound Devices

Audio In: Virtual Cable 1 In

Audio Out: Plantronics headset

Ringing: Crystal WDM Audio

☑ Ring PC speaker

☑ Enable automating sound device settings adjustment

Time to test this virtual connection. You’ll have to manually operate your Greeting Message for this test.

Have a contact call you or set up a second Skype client on your desktop and call yourself. Instructions can be found at SkypeJournal.com.

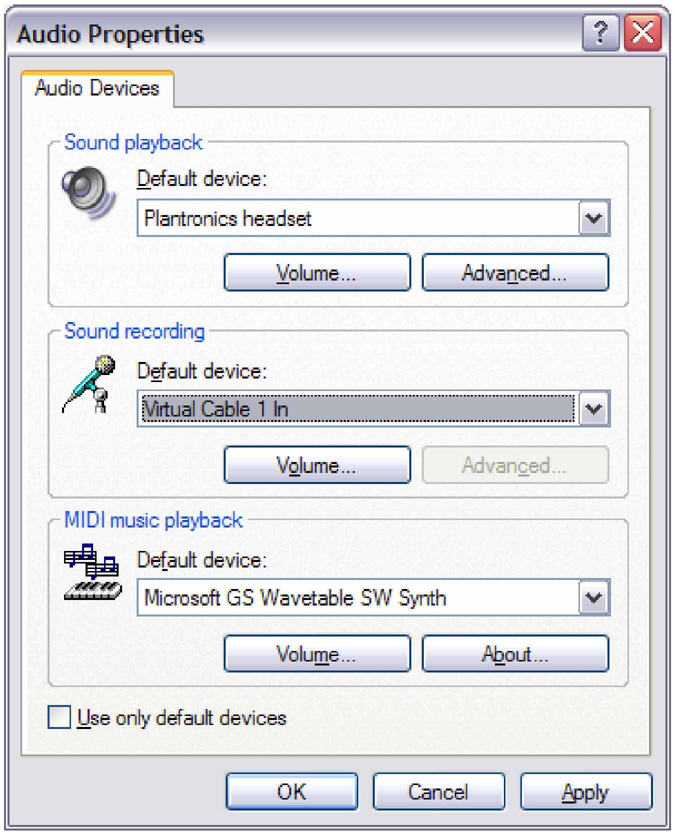
With your Greeting Message file open, answer the call and play your Greeting Message. Voila… it works!

The next step is to configure your Sound Recorder and Skype client to record your caller’s message.

To record your caller’s message, you’ll have to disconnect the first virtual connection and use the Virtual Audio Cable to connect Audio Out from your Skype client to the Sound recording (microphone) in Sound Recorder.

These screen shots illustrate how this is done.

Audio Devices in Sound Recorder:
For Sound playback, select “your speakers” as the Default device.
For Sound recording, select “Virtual Cable 1 In” as the Default device.



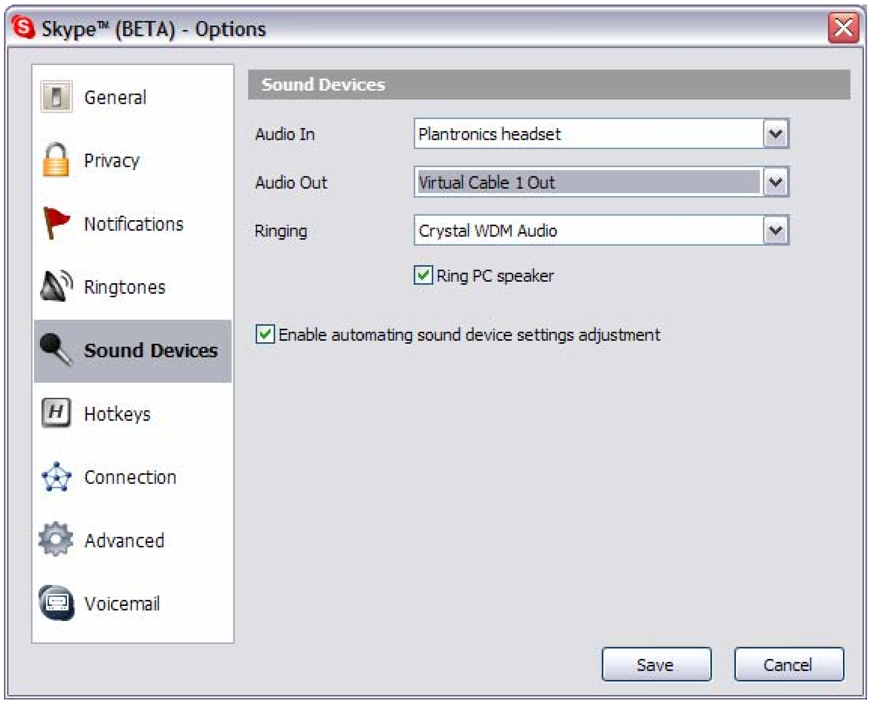
Sound playback
Default device: Plantronics headset

Sound recording
Default device: Virtual Cable 1 In

MIDI music playback
Default device: Microsoft GS Wavetable SW Synth

☐ Use only default devices

Sound Devices in Skype:
For Audio In, select “your microphone”.
For Audio Out, select “Virtual Cable 1 Out”.



Sound Devices

Audio In: Plantronics headset
Audio Out: Virtual Cable 1 Out
Ringing: Crystal WDM Audio
☑ Ring PC speaker
☑ Enable automating sound device settings adjustment

Now you or one of your contacts can place a call to your Skype client, you answer the call and record the message.

Next, you need the Skype API to control this process. The following procedure can actually be executed manually by entering the Skype API Commands into the SkypeTracer Utility. Of course the idea is to later execute this same procedure in a VB.NET program along with a program using Microsoft’s multimedia control to allow you to manage Media Control Interface (MCI) devices such as Windows Sound Recorder.

Place a call to your Skype client. You’ll see in the message window of the SkypeTracer CALL XXXX STATUS RINGING

Answer Call after 10 seconds by issuing the following command through SkypeTracer:
HOOK OFF

Then issue the command:
SET AUDIO_IN Virtual Cable 1 In

Open your Greeting Message file and play it.

Close your recorded Greeting Message file.

Now issue a command:

```
SET AUDIO_IN Plantronics headset
```

Followed by the command:

```
SET AUDIO_OUT Virtual Cable 1 Out
```

Open Sound Recorder

Click Record (button with red circle) on Sound Recorder and leave yourself a message.

Issue the command to hang up the call:

```
HOOK ON
```

Save the Sound file.

Return your Skype client to its normal state with the command:

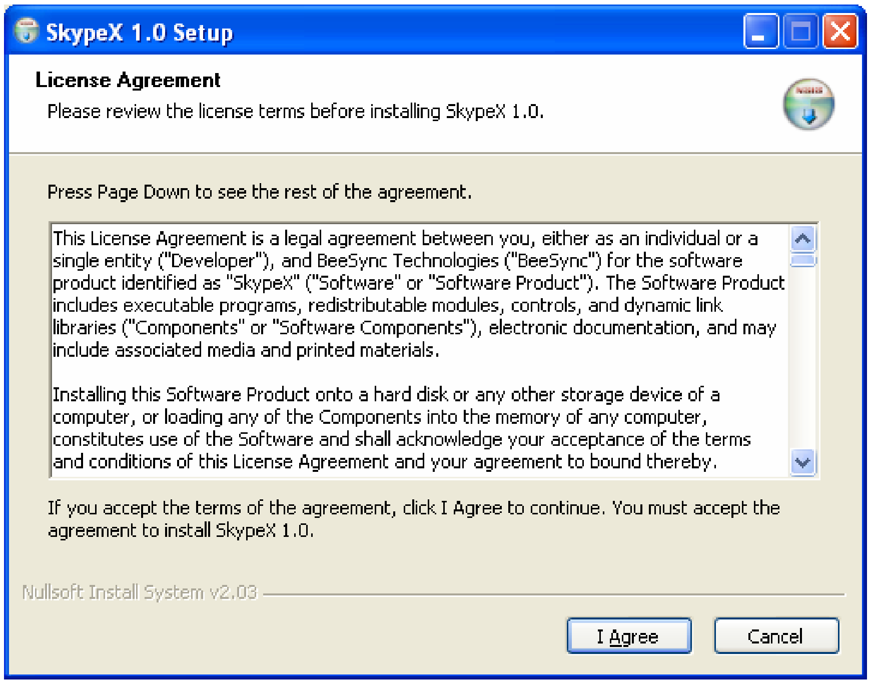
```
SET AUDIO_OUT Plantronics headset
```

Install SkypeX

Next, download and install SkypeX, another Skype API COM wrapper that will also allow you to communicate with the API using Visual BASIC instead of C++.

Download it from beeSync.

The SkypeX COM library comes with an installer/uninstaller program.


SkypeX 1.0 Setup
License Agreement
Please review the license terms before installing SkypeX 1.0.
Nullsoft Install System v2.03
I Agree
Cancel

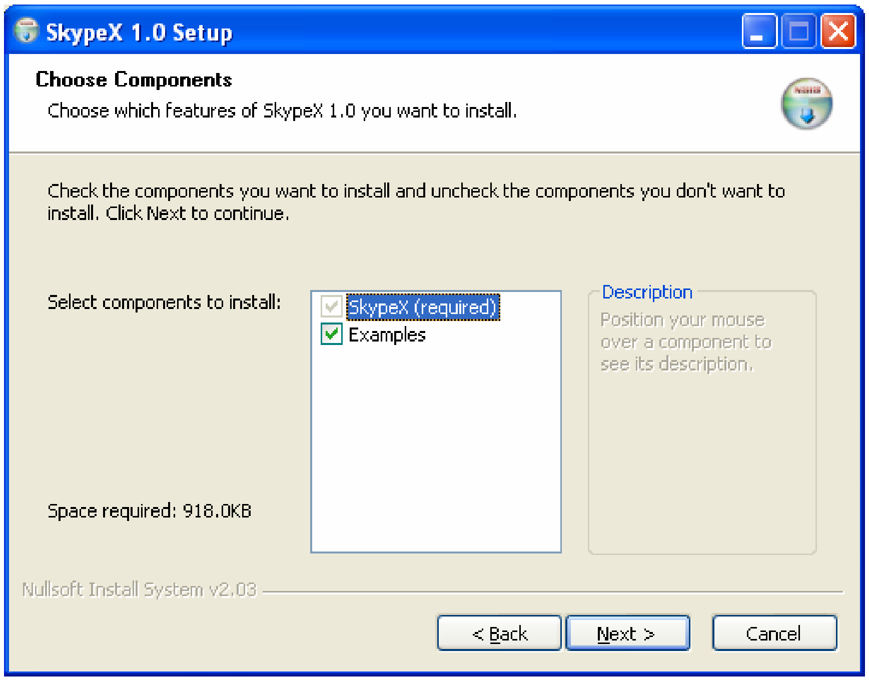
Press Page Down to see the rest of the agreement.

This License Agreement is a legal agreement between you, either as an individual or a single entity ("Developer"), and BeeSync Technologies ("BeeSync") for the software product identified as "SkypeX" ("Software" or "Software Product"). The Software Product includes executable programs, redistributable modules, controls, and dynamic link libraries ("Components" or "Software Components"), electronic documentation, and may include associated media and printed materials.

Installing this Software Product onto a hard disk or any other storage device of a computer, or loading any of the Components into the memory of any computer, constitutes use of the Software and shall acknowledge your acceptance of the terms and conditions of this License Agreement and your agreement to bound thereby.

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install SkypeX 1.0.

Figure 1 - SkypeX End-User License Agreement


SkypeX 1.0 Setup

Choose Components

Choose which features of SkypeX 1.0 you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

- [✓] SkypeX (required)
- [✓] Examples

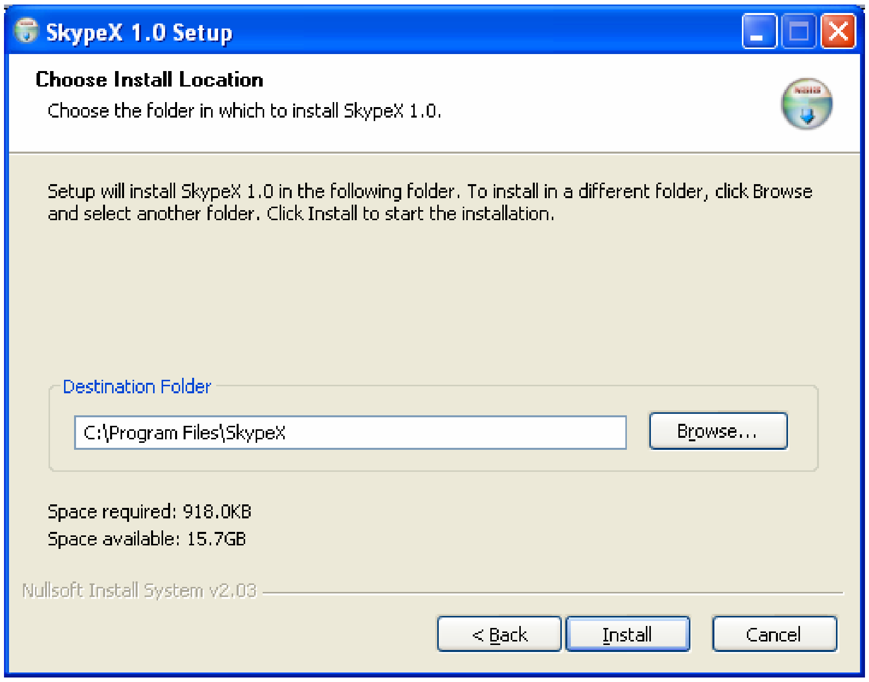
Description

Position your mouse over a component to see its description.

Space required: 918.0KB

Nullsoft Install System v2.03

Figure 2 Select components


SkypeX 1.0 Setup

Choose Install Location

Choose the folder in which to install SkypeX 1.0.

Setup will install SkypeX 1.0 in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder

C:\Program Files\SkypeX

Space required: 918.0KB
Space available: 15.7GB

Nullsoft Install System v2.03

Figure 3 Select Target Folder

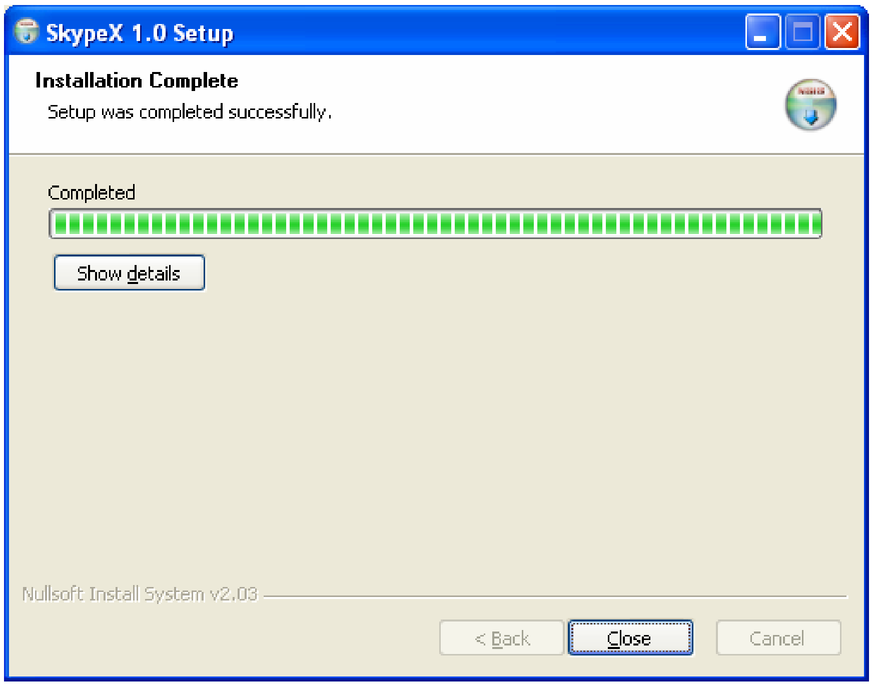


Figure 4 Setup Completed

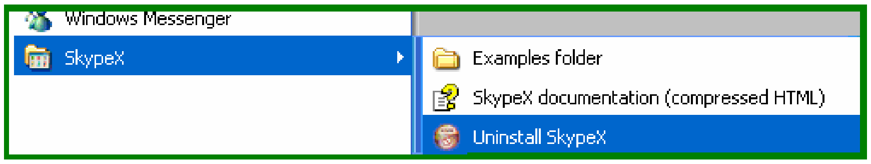
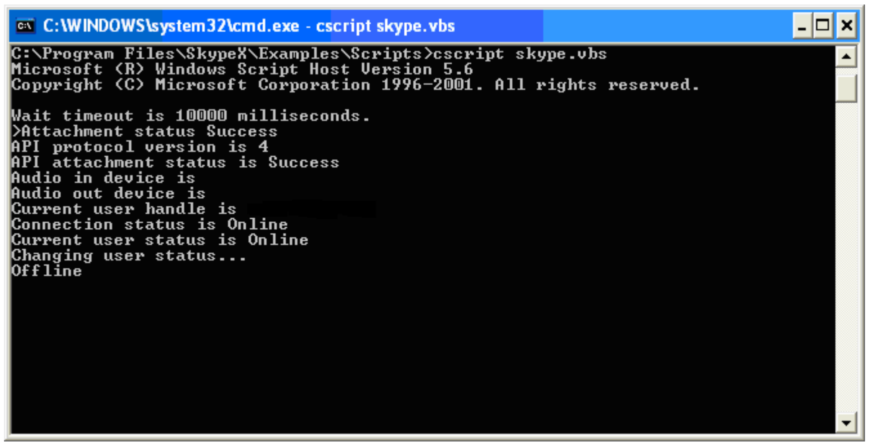


Figure 5 Start Menu Shortcuts

SAMPLE PROGRAMS FOR SKYPEX

SkypeX comes with number of VBScript and HTML examples. They are located in the Examples\Scripts and Examples\Explorer subfolders in the target installation directory.

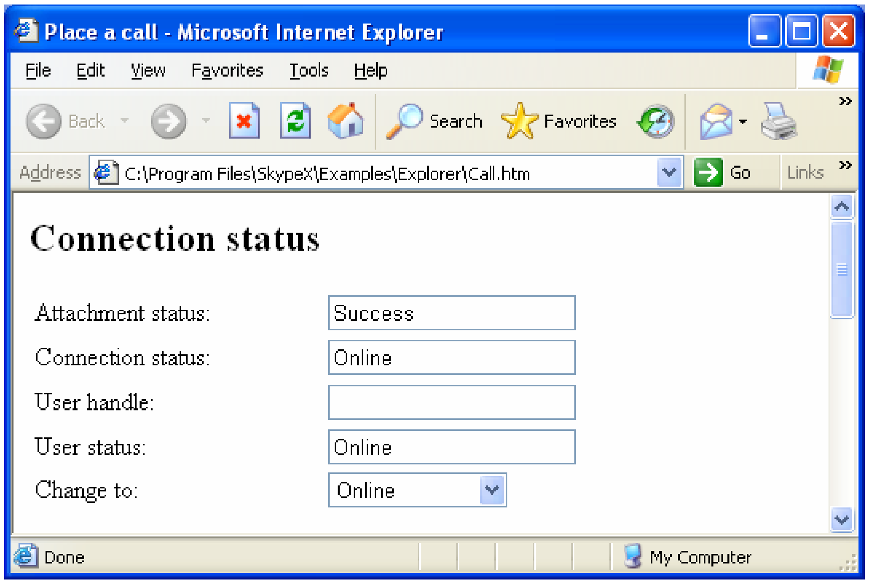
To run the VBScript examples you need the WSH (Windows Scripting Host) command line utility **cscript.exe**.



```
C:\Program Files\SkypeX\Examples\Scripts>cscript skype.vbs
Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.

Wait timeout is 10000 milliseconds.
>Attachment status Success
API protocol version is 4
API attachment status is Success
Audio in device is
Audio out device is
Current user handle is [redacted]
Connection status is Online
Current user status is Online
Changing user status...
Offline
```

Figure 6 - Running the VBScript examples from command line

To run the HTML examples you need the Microsoft Internet Explorer 6.0.


Place a call - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address C:\Program Files\SkypeX\Examples\Explorer\Call.htm
Go
Links
Done
My Computer

Connection status

Attachment status:	Success
Connection status:	Online
User handle:	
User status:	Online
Change to:	Online

Figure 7 Running the HTML example in Internet Explorer

USING SKYPEX FROM VBSCRIPT

```
'// Create Skype object
Set oSkype = WScript.CreateObject("SkypeX.Skype", "Skype_")

oSkype.Timeout = 30000
WScript.Echo "Wait timeout is " & oSkype.Timeout & " milliseconds."

'// Start Skype minimized and without splash screen
oSkype.Start True, True
oSkype.Attach 4
```

USING THE ACTIVEX CONTROL FROM HTML

```
<object classid="clsid:b22983f9-6863-49d9-86ad-0d45fd46298d" name="SkypeCtrl"
width="0" height="0" id="SkypeCtrl">
  <param name="Mute" value="false" />
  <param name="AudioIn" value="" />
  <param name="AudioOut" value="" />
  <param name="CurrentUserStatus" value="1" />
  <param name="Protocol" value="4" />
  <span class="c1">Failed to load SkypeX control.</span>
</object>
```

APPLICATIONS OF SKYPEX

1. Make conference calls.
2. Send Instant Messages.
3. Search for users.
4. Support the Phone API.
5. Send “raw” API commands.

APPENDIX

Source code for Sample Chat Message Utilities in other programming languages.

Visual BASIC.NET Sample Chat Message Utility:

VB6 Source Upgraded by Kevin Delaney (`sillyrabbit999`).

```
Public Class Form1

    Private m_objConversion As SKYPEAPILib.Conversion
    Private WithEvents m_objSkype As
SKYPEAPILib.Access

    Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        ''Do the following when the program loads.
        m_objConversion = New SKYPEAPILib.Conversion

        On Error GoTo NoInit "When there's an error, Go down
to "NoInit".
    m_objSkype = New SKYPEAPILib.Access
        m_objSkype.Connect() "Connect to the Skype API.
        Exit Sub

NoInit: "This is what's going to be done
        MsgBox(Err.Description, MsgBoxStyle.Critical
+ MsgBoxStyle.OKOnly, "Unable to Connect to Skype")
"Make a messagebox with the error happens.
    Me.Close() "Close the program.
    End Sub

  Private Sub SendCommand_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
SendCommand.Click
        Dim s As String

        On Error GoTo NoCommand "When there's an error, Go
down to "NoCommand".
```

```
        s = "MESSAGE " & handle.Text & " " &
message.Text "Set raw message to Skype in format "MESSAGE
<handle> <chat message>"
    If Len(s) = 0 Then
            Exit Sub
        End If
        m_objSkype.SendCommand(s) "Send the command
        Exit Sub

NoCommand:
        MsgBox(Err.Description, MsgBoxStyle.Critical
+ MsgBoxStyle.OKOnly, "Unable to Send Command") "Make a
messagebox with the error happens.
    End Sub

End Class
```

Perl Sample Chat Message Utility:

Following is an Example of how to create Skype Chat Graphical User Interface in Perl using TK package and Skype API Com Wrapper.

This is an example application to demonstrate the use of the Skype API and Perl. This example uses Perl TK across platform GUI toolkit for Perl, which means this program will run under Windows, Linux, Mac, etc., with only a small amount of modification (for Skype API calls).

GETTING STARTED

In order to get started you should do following (if you haven't already):

1. Download and install ActivePerl for Windows (MSI) from ActiveState.com.
2. Download and install Skype API Com Wrapper from KhaosLabs.com.
3. Run example from its directory as Perl perlforskype.pl

REFERENCE DOCUMENTATION

1. Skype's API Documentation
2. Perl for Windows
3. Perl TK

Note: If you want to study Skype-related stuff only go to line 316 or SndMsg() function

SOURCE CODE

```
use Tk;
use Tk::widgets ;
use subs qw/build_menubar fini init/;
use vars qw/$MW $VERSION/;
use strict;
```

#Perl uses Win32::OLE module to interact with COM in Windows

```
use Win32::OLE qw(in);

use constant vbCr => "\r";
```

1. Create a New Window
2. Set the Window Size

3. Set the title of the window

```
$MW = MainWindow->new;
$MW->geometry("359x283+8+8");
init;
MainLoop;

sub build_menubar {
```

In this function we are creating a menu bar:

- Create Menu File and Help
- Create the menu items for each menu.
- n, ne, nw, s, sw, se, e, w are all directions. n = North, s = South and so on.
- text = The text displayed in the label
- background = label's background color
- cursor = Specifies the mouse cursor to be used for the widget. The value may have any of the forms acceptable to `Tk_GetCursor`
- font = Font of the text in the label
- foreground = This is the font color

# Create the menu bar and File and Quit menu buttons. Note that the cascade's menu widget is automatically created.

```
my $menubar = $MW->Menu;
$MW->configure(-menu => $menubar);
my $file = $menubar->cascade(-label => '~File');
my $help = $menubar->cascade(-label => '~Help',-tearoff => 0);
```

# Create the menu items for each menu. First, the File menu item.

```
$file->command(-label => "Quit", -command => \&fini);
```

# Finally, the Help menu items.

```
$help->command(-label => 'Version');
$help->separator;
$help->command(-label => 'About');

my $ver_dialog =   $MW->Dialog(-title   => 'PaGuX Perl Skype Chat',
      -text    => "PaGuX Perl Skype Chat\n\nVersion $VERSION",
      -buttons => ['OK'],
      -bitmap  => 'info');

my $about_dialog = $MW->Dialog(-title   => 'About PaGuX Perl Skype Chat',
                 -text    => 'PaGuX Perl Skype Chat',
                 -buttons => ['OK']);

my $menu = $help->cget('-menu');
$menu->entryconfigure('Version', -command => [$ver_dialog   => 'Show']);
$menu->entryconfigure('About',   -command => [$about_dialog => 'Show']);

$menubar;                  # return the menu bar

 } # end build_menubar


sub build_elements {
```

In this function we build all the Controls on Form or Main window

1. Labels
2. Create Input or entry box for Skype Handle or Message
3. Create a Command button which fires & passes Skype Handle and Message to SndMsg function

```
my $hndl;
my $msg;
```

Now we are creating a Label

- anchor = the position of the Text inside the Label, options are:
    - n, ne, nw, s, sw, se, e, w are all directions. n = North, s = South and so on.
    - text = The text displayed in the label
    - background = label's background color
    - cursor = Specifies the mouse cursor to be used for the widget. The value may have any of the forms acceptable to `Tk_GetCursor`
    - font = Font of the text in the label
    - foreground = This is the font color

```
my $label = $MW->Label(
                  -anchor => "w",
                  -text => "Enter a Skype Handle:",
                  -background => "#D4D0C8",
                  -cursor => "",
                  -font => "Tahoma 9 bold",
                  -foreground => "#000077"
                  )->pack;
```

Now we are going to specify where to place the label we just created on to the form, so we set the width and height of the label and give x/y coordinates of where to place it at.

```
$label->place(
                  -width => 160,
                  -height => 16,
                  -x => 8,
                  -y => 50
                  );
```

Now we are going to create an Entry box also known as TextBox or InputBox. We will this to grab the input to check if the inputted date string is in valid format.

```
my $txtInput = $MW->Entry(
                  -textvariable => \$hndl,
                  -borderwidth => 1,
                  -cursor => "",
                  -font => "Tahoma 8 normal",
                  -foreground => "#000000",
                  -relief => "sunken"
                  )->pack;

$txtInput->place(
                  -width => 152,
                  -height => 24,
                  -x => 155,
                  -y => 50
                  );

my $lblDisplay = $MW->Label(
                  -anchor => "w",
                  -text => "Enter your Message : ",
                  -background => "#D4D0C8",
                  -cursor => "",
                  -font => "Tahoma 9 bold",
                  -foreground => "#000077"
                  );
$lblDisplay->place(
                  -width => 125,
                  -height => 24,
                  -x => 8,
                  -y => 120
                  );

my $msgInput = $MW->Entry(
                  -textvariable => \$msg,
                  -borderwidth => 1,
                  -cursor => "",
                  -font => "Tahoma 8 normal",
                  -foreground => "#000000",
                  -relief => "sunken"
                  )->pack;

$msgInput->place(
                  -width => 135,
                  -height => 24,
                  -x => 155,
                  -y => 120
                  );
```

Next we create a Button, also known as a CommandButton. This is what we will use to call the IsDate Sub when this button is clicked on

- command = This calls the sub. sub{sub_name_here(input1, input2)} and so on, you can have more than one input or have no inputs.
- Without arguments:
    - command => `\&subname`
    - command => `sub { ... }`
    - command => `'methodname'`
- Or with arguments:
    - command => `[ \&subname ?, args ...? ]`
    - command => `[ sub { ... } ?, args...? ]`
    - command => `[ 'methodname' ?, args...?]`

```
my $cmdSndMsg = $MW->Button(
            -activebackground => "#FFFCBF",
            -activeforeground => "#E30229",
            -background => "#FFFFFF",
            -borderwidth => 1,
            -text => "Send Message",
            -command => sub{SndMsg($hndl,$msg)},
            -cursor => "",
            -font => "Tahoma 8 bold",
            -foreground => "#140F7B",
            -relief => "solid"
            )->pack;

# Place the button on the form
$cmdSndMsg->place(
                  -width => 104,
                  -height => 24,
                  -x => 170,
                  -y => 160
            );

my $lblError = $MW->Label(
                  -anchor => "w",
                  -borderwidth => 1,
                  -text => "",
                  -background => "#FFFFFF",
                  -cursor => "",
                  -font => "Tahoma 8 normal",
                  -relief => "solid",
                  -highlightbackground => "#000000",
                  -justify => "right"
            )->pack;

our $lblError = $MW->Label(
                  -anchor => "w",
                  -borderwidth => 1,
                  -text => "",
                  -background => "#D4D0C8",
                  -cursor => "",
                  -font => "Tahoma 8 normal",
                  -justify => "right"
            )->pack;


$lblError->place(
                        -width => 328,
                        -height => 44,
                        -x => 12,
                        -y => 190
                  );

    }
```

Next, place all the subs below here. We only have one sub so it's below. But if we had more than one sub, then it could go anywhere just as long as it's below the `MainLoop`;

```
sub SndMsg() {
```

This function is called when user clicks on the Command button on the form. Following in this function, we:

1. Get Skype Handle and Message parameters.

   ```
   my $txtInput=$_[0];
   my $msgInput=$_[1];
   ```

2. Check if parameters are empty or not.

```
my $error_message = "";

$error_message .= "Please enter a Skype Handle " if (
!$txtInput );
$error_message .= "Please specify your Message " if (
!$msgInput );

if ( $error_message )
 {
  our $lblError->configure(-text => $error_message );
 }
 else
 {
```

3. If parameters not empty we initiate a new OLE object of type SKYPEAPI.Access.

```
our $lblError->configure(-text => "" );
my $objSkype;
$objSkype = Win32::OLE->new('SKYPEAPI.Access',
'objSkype_');
```

4. Connect to Skype Network.

```
   $objSkype->Connect();
```

5. Use SendMessage function of Skype API to send a chat message as variables supplied by user.

```
   $objSkype->SendMessage($txtInput, $msgInput);

    }

    }

 sub fini {

     exit;

 } # end fini


sub init {
```

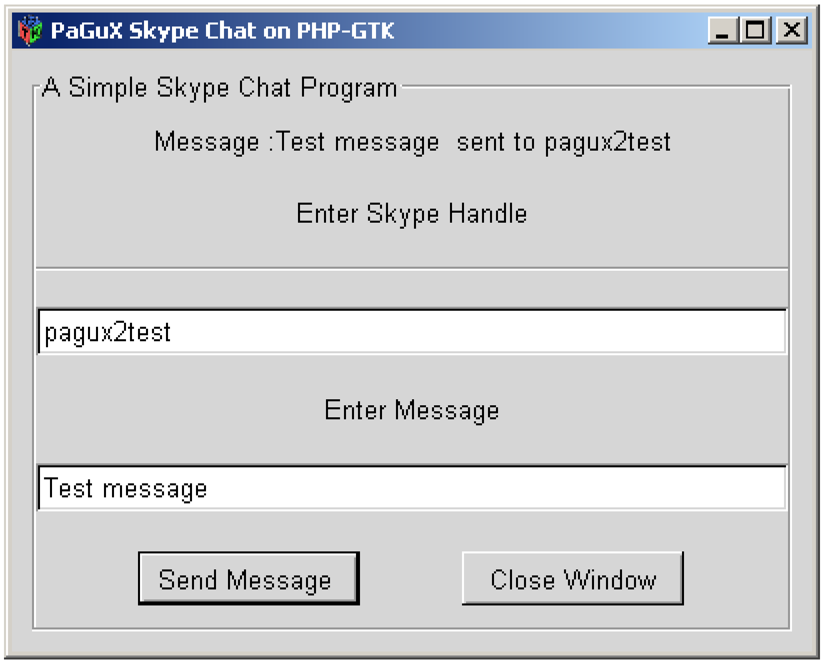
In this function we initialize

1. Title and some other variables
2. Call build_menubar to build menu bar
3. Call build_elements to build controls on Main window

```
$VERSION = '1.0';

$MW->title("PaGuX Perl Skype Chat $VERSION");

my $menubar = build_menubar;

build_elements;

# my $frame = $MW->Frame(qw/-width 400 -height 350 )-
>pack;

 } # end init
```

PHP GTK Sample Chat Message Utility:

ABOUT



Message :Test message sent to pagux2test

Enter Skype Handle

pagux2test

Enter Message

Test message

This is a sample application to demonstrate the use of the Skype API and PHP (Hypertext Preprocessor) scripting language. This example uses PHP GTK on a cross platform GUI toolkit for PHP, which means this program will run under windows, Linux, Mac, etc., with only small amount modification (for Skype API calls).

GETTING STARTED

In order to get started you should do following

1. Download & install php-gtk for Windows from here.
2. Download and install Skype API Com Wrapper from KhaosLabs.com.
3. Run example from its directory as php phpforskype.php

Add zlib to folder…..

REFERENCE DOCUMENTATION

1. Skype's API Documentation
2. PHP COM for Windows
3. PHP-GTK

Note: If you want to study Skype-related stuff only go to line 90 or SndMsg() function.

SOURCE CODE

/* This “if” statement determines where it finds the appropriate file to run the gtk program. It is for Windows and Linux (respectively) */

```
if (!class_exists('gtk'))
{
      if (strtoupper(substr(PHP_OS, 0, 3)) == 'WIN')
                  dl('php_gtk.dll');
            else
                  dl('php_gtk.so');
}
```

/* This function closes the window. It is defaulted to false. */

```
function delete_event()
{
      return false;
}
```

/* This is very important to actually destroying any instances of PHP running. */

```
function destroy()
{
      Gtk::main_quit();
}
```

/* This takes the instance of entry class and grabs what has been entered into the textbox and saves it to $gettext. Then the parameters are passed to SendMessage function. */

```
function send_message($label, $entry,$message)
{
$gethandle = $entry->get_text();
$getmessage = $message->get_text();

if((strlen($gethandle))==0 ||
(strlen($getmessage))==0)
            {
            $sentmsg ="Handle or Message cannot be
            empty, pl fill it";
            $label->set_text($sentmsg);
}else
{
```

/* We now create a new instance of COM object SKYPEAPI.Access. Use Connect API call. Use SendMessage API function to send Message. */

```
//com_load_typelib("SKYPEAPILib.Conversion");
```

```
$com = new COM("SKYPEAPI.Access") or die("can not
create SKYPEAPILib.Access object");

$com->Connect();

$com->SendMessage($gethandle, $getmessage);

      $sentmsg ="Message :".$getmessage." sent to
      $gethandle";
      $label->set_text($sentmsg);
      }
}
/* Create the window. */
$window = &new GtkWindow();
$window->set_name('main window');
$window->set_title('PaGuX Skype Chat for PHP-GTK');
$window->set_usize(400, 300);
$window->connect('destroy', 'destroy');
$window->connect('delete-event', 'delete_event');
$window->set_border_width(10);
$window->set_position(GTK_WIN_POS_CENTER);
/* Create the frame. */
$frame = &new GtkFrame('A Simple Skype Chat
Program');
$window->add($frame);
/* Create the vertical box for putting things in. */
$box1 = &new GtkVBox();
$frame->add($box1);
$box1->show();
/* Create the area where will be updating the information. */
$label = &new GtkLabel('');
$box1->pack_start($label);
$label->show();

$labelhandle = &new GtkLabel('Enter Skype Handle');
      $box1->pack_start($labelhandle);
      $labelhandle->show();
  /* Create a horizontal line. */
$separator = &new GtkHSeparator();
$box1->pack_start($separator);
$separator->show();
```

```
/* Create the textbox for the user to enter in the information. */
$entry = &new GtkEntry();
$box1->pack_start($entry);
$entry->show();

$labelmsg = &new GtkLabel('Enter Message');
      $box1->pack_start($labelmsg);
      $labelmsg->show();
      $box1->pack_start($separator);
      $separator->show();

      $message = &new GtkEntry();
            $box1->pack_start($message);
      $message->show();

/* Create the horizontal box for across the bottom of the window. */
$box2 = &new GtkHButtonBox();
$box2->set_layout(GTK_BUTTONBOX_SPREAD);
$box1->add($box2);
$box1->show();

/* Set up one button that is in the button box. */
$button = &new GtkButton('Send Message ');
$button->connect_object('clicked', 'send_message',
$label, $entry,$message);
      $box2->pack_start($button);
      $button->show();

/* Set up the other button in the button box */
$button = &new GtkButton('Close Window');
$button->connect_object('clicked', 'destroy');
$box2->pack_start($button);
$button->show();

/* Close off the frame by showing it. */
$frame->show();
/* Tell the window to show all elements */
$window->show_all();

/* Finish off the entire program. */
Gtk::main();
?>
```

Java Sample Chat Message Utility:

INTRODUCTION

This section intends to give a practical guide on how to use Java language to access the Skype API, which is originally targeted for C/C++ programmers. This guide is based on *JSkype* – a Java Native Interface (JNI) implementation to enable Java clients to use the Skyp API. JSkype is provided by Bart Lamot and now is evolving into an open source (LGPL license) project namely *Java to Skype API* (JSA), accessible at SourceForge.net. The new project plans to provide new interface to support all parts of the API but requires no knowledge of the original Skype API itself which means some abstract layers (probably some classes hierarchy) will be composed to encapsulate and hide the actual calls of Skype API.

Unfortunately, there is no new release existent (as of April 9, 2005) at its new home site and the only available version is still on its previous site. It would have been nice if there could be more instructions on its old site, as it is actually still working well if you set it up correctly by yourself (almost like hacking), although the new site has also disclaimed the validity of the old implementation.

INSTALLATION

These step-by-step instructions, according to my experience, on how to setup the JSkype JNI should save at least half a day of your time!

1. Download JSkype from its old site. It includes both the binary and source codes in C and Java. Unpack it to your local directory and you should see JSkype.jar and JSkype.dll under the extracted sub directories.

2. The JNI (.dll) file is compiled in MS VC++ 6.0, and you need to download an extra .dll file – Microsoft C Runtime Library (v. 6.0.8337.0) here. The lack of this runtime library is not intuitive to figure out and it is not mentioned on JSkype site.

3. If you want to run the Java example within the JSkype package without any modification, you would also have to download **Standard Widget Toolkit** (*SWT*) from Eclipse. However, you only need two

files (swt-win32-30xx.dll and swt.jar), the xx states for its version label which the latest by the writing of this guide is 64.

4. Put these files in the current path and then you can run the example application in the JSkype. To save your efforts of exploring how to specifying Java CLASSPATH on your own, I have packed everything aforementioned so that you can go straight to see the example running by just unpacking all the files and run the batch files. Get the whole pack here.

5. This step is needed only if you have not installed **Java 2 Platform** (*JDK*) **or Java Runtime Environment** (*JRE*) later than v1.4.2. Then, you can download a copy of stripped JRE from here.

RUNNING THE JSKYPE EXAMPLE

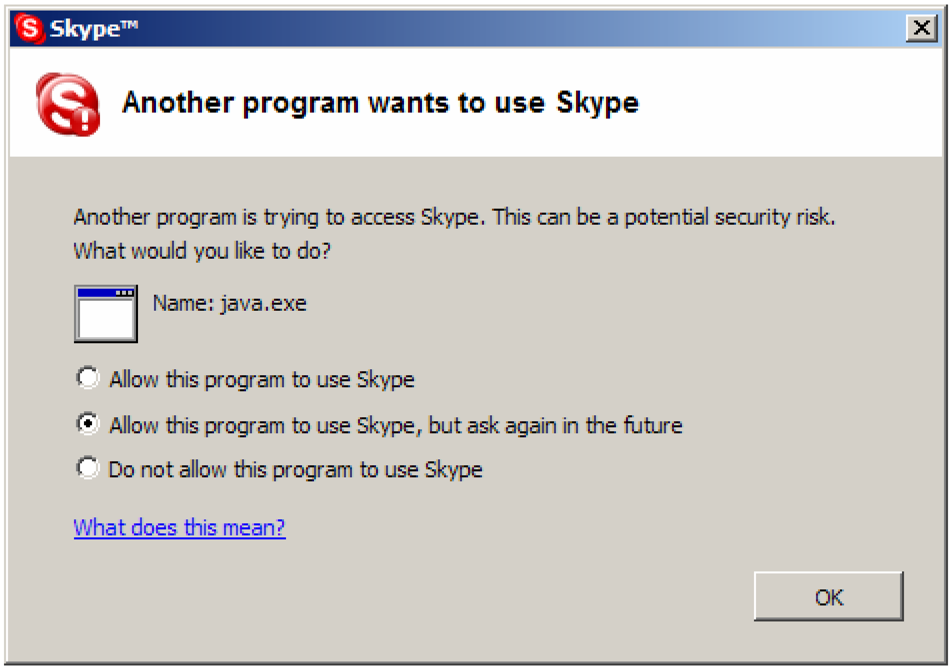
If you have **Java 2 Platform** (JDK/JRE v1.4.2 or later) installed, then after the installation steps above, you only need to run the batch file JSkypeExample.bat, which has the following content:

```
REM set path = .\jre\bin,%path%
java -classpath .;jskype.jar;swt.jar
net.lamot.java.jskype.swtclient.swtChatWindow
```

If no Java installed, you need to extract the JRE.zip to your JSkype directory and then uncomment the first line in the batch file:

```
SET PATH = .\JRE\BIN,%PATH%
java -classpath  .;jskype.jar;swt.jar
net.lamot.java.jskype.swtclient.swtChatWindow
```

You will then see a Skype warning window


Skype™

Another program wants to use Skype

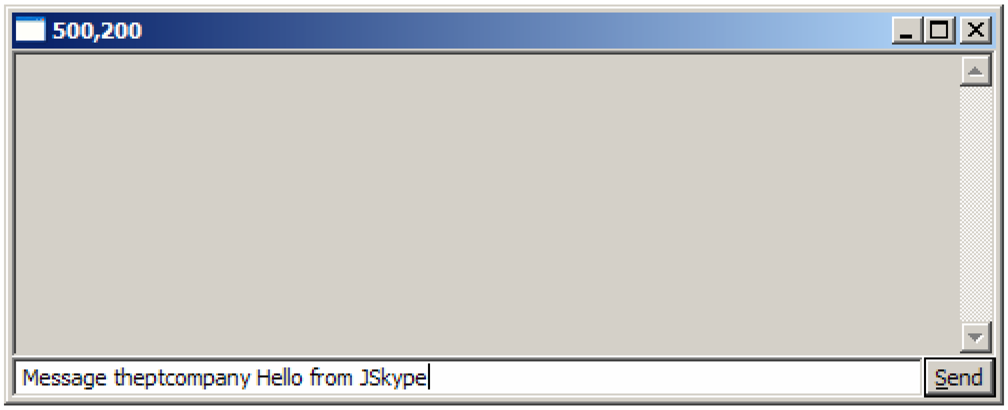
Another program is trying to access Skype. This can be a potential security risk. What would you like to do?

Name: java.exe

( ) Allow this program to use Skype
[✓ Allow this program to use Skype, but ask again in the future]
( ) Do not allow this program to use Skype

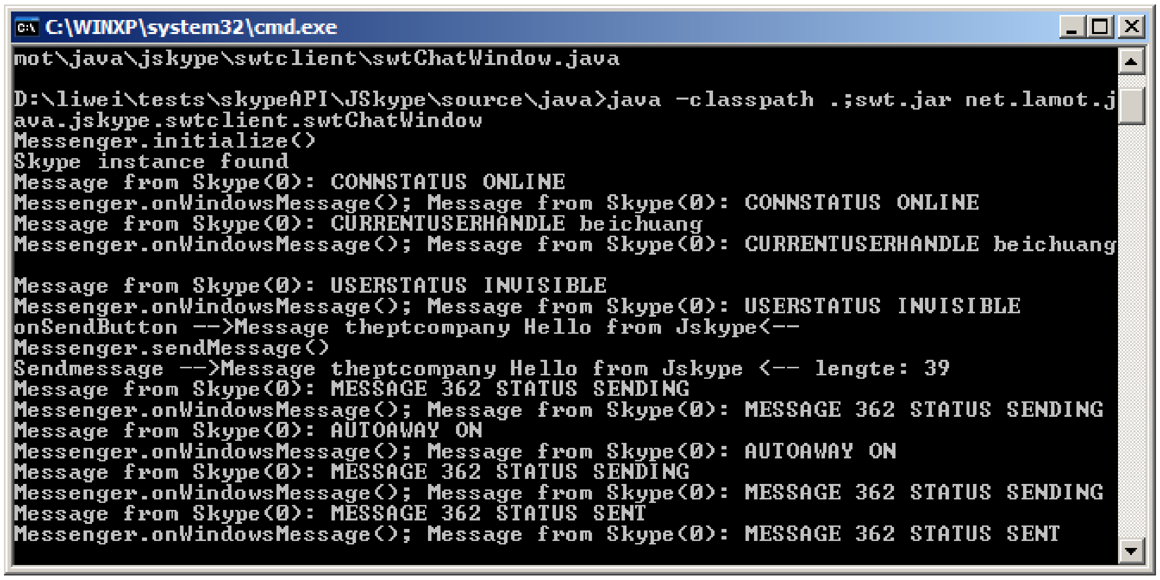
What does this mean?

Click OK button to continue, fill some command in the text box and client Send



Message theptcompany Hello from JSkype

And on the prompt window you will see lines like:



```
mot\java\jskype\swtclient\swtChatWindow.java

D:\liwei\tests\skypeAPI\JSkype\source\java>java -classpath .;swt.jar net.lamot.j
ava.jskype.swtclient.swtChatWindow
Messenger.initialize()
Skype instance found
Message from Skype(0): CONNSTATUS ONLINE
Messenger.onWindowsMessage(); Message from Skype(0): CONNSTATUS ONLINE
Message from Skype(0): CURRENTUSERHANDLE beichuang
Messenger.onWindowsMessage(); Message from Skype(0): CURRENTUSERHANDLE beichuang

Message from Skype(0): USERSTATUS INVISIBLE
Messenger.onWindowsMessage(); Message from Skype(0): USERSTATUS INVISIBLE
onSendButton -->Message theptcompany Hello from Jskype<--
Messenger.sendMessage()
Sendmessage -->Message theptcompany Hello from Jskype <-- lengte: 39
Message from Skype(0): MESSAGE 362 STATUS SENDING
Messenger.onWindowsMessage(); Message from Skype(0): MESSAGE 362 STATUS SENDING
Message from Skype(0): AUTOAWAY ON
Messenger.onWindowsMessage(); Message from Skype(0): AUTOAWAY ON
Message from Skype(0): MESSAGE 362 STATUS SENDING
Messenger.onWindowsMessage(); Message from Skype(0): MESSAGE 362 STATUS SENDING
Message from Skype(0): MESSAGE 362 STATUS SENT
Messenger.onWindowsMessage(); Message from Skype(0): MESSAGE 362 STATUS SENT
```

Basically, this example works as a Java version of SkypeTracer or the msgapitest example in the official Skype API. You can test with more Skype API command e.g., Call, SET AUDIO_IN etc.

BUILDING YOUR JAVA APPLICATION USING JSKYPE

This section gives you some idea how to use the JSkype in your Java program. The code resemble the demo above to illustrate shows how to send text message to a Skype user with name “theptcompany”.

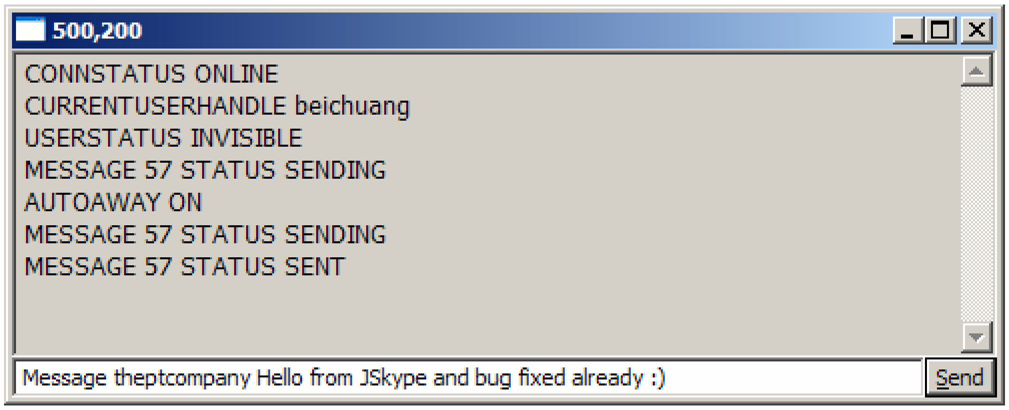
```
// Import the JSkype packages
import net.lamot.java.jskype.general.AbstractMessenger;
import net.lamot.java.jskype.general.MessageListenerInterface;
import net.lamot.java.jskype.windows.Messenger;
import java.lang.Thread;
import java.lang.Exception;

// The MessageListenerInterface interface defines a method
onMessageReceived suppose to be called when notifications are
received from run-time Skype Client.
public class JSkypeTest implements MessageListenerInterface {

// Declare a messenger object to be used to send Skype commands
private AbstractMessenger msgr = null;
// Create a new instance of JSkypeTest.
    public JSkypeTest() {
            try {
        // pause 6 seconds to wait for the
initialization of JSkype
              Thread.sleep(6000);
        // send the Skype API text command
              msgr.sendMessage("Message theptcompany
hello from JSkypeTest" );
      }
      catch (Exception e) {
              e.printStackTrace();
      }
    }

//  * @param args the command line arguments
    public static void main(String[] args) {
        new JSkypeTest();
    }

     public void onMessageReceived(String str) {

// We simply just print out if any notifications arrive through JSkype
      System.out.println(str);
    }

}
```

The code above can be compiled with the batch file JSTest.bat, which has the content

```
      javac –classpath .;JSkype.jar JSkypeTest.java
      java –classpath .;JSkype.jar JSkypeTest
```

However, if you are using the original JSkype package, you will soon notice that the method `onMessageReceived` is never called. It means your Java program will not receive any information from the JSkype, which in turn means you can only send a command to JSkype but it's not possible to receive any feedback from the runtime Skype client to determine whether your Text Message has been sent successfully to the receiver. This is simply because there's a bug in the original JSkype package. The good thing is that everyone is able to fix it due to its open source nature.

I have fixed such a bug during this writing. So if you run the JSkype example shipped with this guide, you should see the Skype notifications print out not only in the Dos Prompt Window but also in the application GUI. Now you are able to utilize the Skype feedback in your application to conduct more interesting behaviors, e.g., to play a greeting message when your buddies calling in and then record their voice message.



```
CONNSTATUS ONLINE
CURRENTUSERHANDLE beichuang
USERSTATUS INVISIBLE
MESSAGE 57 STATUS SENDING
AUTOAWAY ON
MESSAGE 57 STATUS SENDING
MESSAGE 57 STATUS SENT
```

```
Message theptcompany Hello from JSkype and bug fixed already :)
```

S U M M A R Y

JSkype gives a simple way to hook your Java application with Skype, and you are able to make more use of the Skype network for your development. The Java Native Interface JSkype is rather simple but reasonable robust. However, some methods such as `destroy`, seem not to be implemented. That is why you have to call `system.exit()` to force your application to close. If you cannot wait for the new release from the JSA project, you can look into the source code and complete it yourself. One big advantage with this current release is its simplicity. Enjoy your coding!

Tip: Thinking of *Why* is sometimes more important than just figuring out *How*.

Programming Guide for Microsoft C#

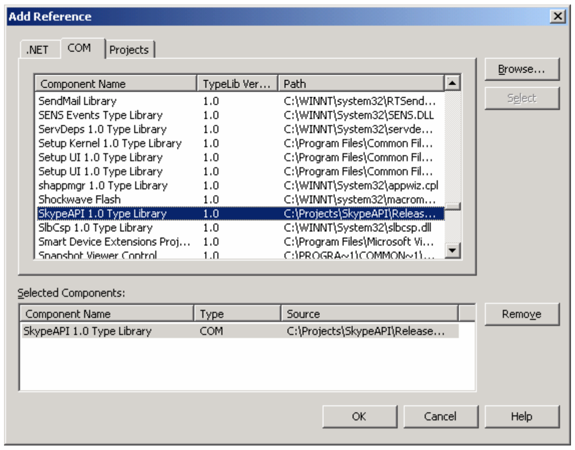
This section demonstrates techniques for utilizing the Skype API COM wrapper in your C# application. Like any .NET language, there is fairly straightforward COM interoperability. About the hardest part is setting up the event handlers.

The basic steps for utilizing the Skype API wrapper include:

1. Adding a reference to the Skype API wrapper library
2. Declaring the Skype interface objects
3. Define event handlers
4. Create objects
5. Implement calls

ADDING A REFERENCE TO SKYPE API WRAPPER LIBRARY

To add a reference to the Skype API library to a C# project, pull down the Project menu, then Add Reference. From the COM tab, select the Skype API library, click on the Select button, then OK.



Component Name	TypeLib Ver...	Path
SendMail Library	1.0	C:\WINNT\system32\RTSend...
SENS Events Type Library	1.0	C:\WINNT\System32\SENS.DLL
ServDeps 1.0 Type Library	1.0	C:\WINNT\System32\servde...
Setup Kernel 1.0 Type Library	1.0	C:\Program Files\Common Fil...
Setup UI 1.0 Type Library	1.0	C:\Program Files\Common Fil...
Setup UI 1.0 Type Library	1.0	C:\Program Files\Common Fil...
shappmgr 1.0 Type Library	1.0	C:\WINNT\System32\appwiz.cpl
Shockwave Flash	1.0	C:\WINNT\system32\macrom...
SkypeAPI 1.0 Type Library	1.0	C:\Projects\SkypeAPI\Releas...
SlbCsp 1.0 Type Library	1.0	C:\WINNT\System32\slbcsp.dll
Smart Device Extensions Proj...	1.0	C:\Program Files\Microsoft Vi...
Snapshot Viewer Control	1.0	C:\PROGRA~1\COMMON~1\...

Component Name	Type	Source
SkypeAPI 1.0 Type Library	COM	C:\Projects\SkypeAPI\Release...

DECLARE SKYPE INTERFACE OBJECTS

In your form (or other implementation class), define two member variables such as:

```
    private ConversionClass m_objConversion;
    private AccessClass m_objAccess;
```

DEFINE EVENT HANDLERS

You can implement event handlers in the same way you would normally handle C# events. You'll need to declare an event handler using the appropriate parameters. You can refer to either the Skype API COM Wrapper documentation or use the Intellisense capability in Visual Studio to view parameters. Here are a couple of examples of function declarations and the calls to monitor those events.

```
m_objAccess.APIStatusChanged +=
new_IAccessEvents_APIStatusChangedEventHandler(APISta
tusChanged);

void APIStatusChanged(SkypeAPIAttachmentStatus
Status)
{
...
}

m_objAccess.MessageReceived += new
_IAccessEvents_MessageReceivedEventHandler(MessageRec
eived);

void MessageReceived(SKYPEAPILib.Message
ReceivedMessage)
{
...
}
```

CREATE OBJECTS

To create the Skype objects previously declared and connect to Skype, you can implement the following code:

```
try
{
// Initialize our objects
m_objConversion = new SKYPEAPILib.ConversionClass();
m_objAccess = new SKYPEAPILib.AccessClass();
// Set up event delegates
m_objAccess.APIStatusChanged += new
_IAccessEvents_APIStatusChangedEventHandler(APIStatus
Changed);
// Trigger a connection to Skype
m_objAccess.Connect();
} catch (Exception ex) {
MessageBox.Show(this, ex.Message, "Unable to
Initialize Skype Connectivity", MessageBoxButtons.OK,
MessageBoxIcon.Error);
}
```

Clean-up is automatic.

IMPLEMENT CALLS

Once you’ve set everything up, you can easily do things like the following:

Iterate through your Skype friend list:

```
foreach(User objUser in m_objAccess.GetFriendList())
{
AddUserHandleToMyList(objUser.Handle)
}
```

Send a message:

```
m_objAccess.SendMessage("MyFriend", "How are You");
```

Change your online status:

```
m_objAccess.CurrentUserStatus =
SkypeOnlineStatus.olsAway;
```

Hang-up all active calls:

```
foreach(Call objCall in
m_objAccess.GetActiveCallList())
{
objCall.Status = SkypeCallProgress.prgFinished;
}
```

Programming Guide for Microsoft C++

This section demonstrates techniques for utilizing the Skype API COM Wrapper in your C++ application. Microsoft provides the Active Template Library (ATL) for C++ beginning with version 6.0 and continuing through the .NET versions. This guide is not a tutorial for ATL; you should be familiar with using the framework and COM in general. The example code shown here can be included in either a Microsoft Foundation Classes (MFC) as well as a "pure-ATL" application.

The basic steps for utilizing the Skype API wrapper include:

1. Importing the type library
2. Implement the IDispEventImpl interface
3. Define event handlers
4. Create Objects
5. Using the CSkypeMessageQueue Class

IMPORT THE TYPE LIBRARY

You'll need to import the library to get all of the data types and proxy classes generated for you. In general, you can put in the following line into `stdafx.h`

```
      #import "c:\\winnt\\system32\\skypeapi.dll"
named_guids
      using namespace SKYPEAPILib;
```

Substitute the actual path where you're storing the skypeapi.dll, and don't forget the double backslashes!

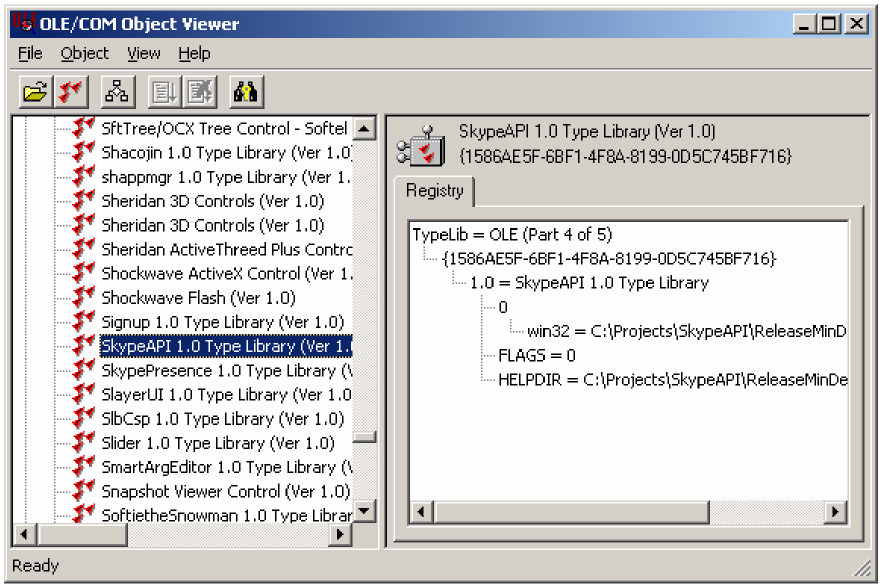
IMPLEMENT THE IDISPEVENTIMPL INTERFACE

There is an ATL mechanism, IDispEventImpl, that allows a class to act as an IConnectionPoint event sink. You can implement this interface in any class, including that of a dialog or window. For example, if you're deploying an MFC Dialog application, you could define:

```
// IDC_MYSKYPE can be any arbitrary unique constant
#define IDC_MYSKYPE 0x100

class CMySkypeAppDlg: public CDialog,
            public IDispEventImpl<IDC_MYSKYPE,
CMySkypeAppDlg,
      &DIID__IAccessEvents, &LIBID_SKYPEAPILib, 1, 0>
{

public:

// Include whatever other MFC/ATL stuff you need

BEGIN_SINK_MAP(CMySkypeAppDlg)
END_SINK_MAP()

protected:

IAccessPtr m_ptrAccess;
IConversionPtr m_ptrConversion;

void ConnectToSkype();
void DisconnectFromSkype();
};
```

DEFINE EVENT HANDLERS

For each callback event that you'll be handling, you need to define a function properly formatted. You can use the **OLE View** utility to see the function prototypes for the IAccessEvents class. In the Type Libraries list, double click on the SkypeAPI library.



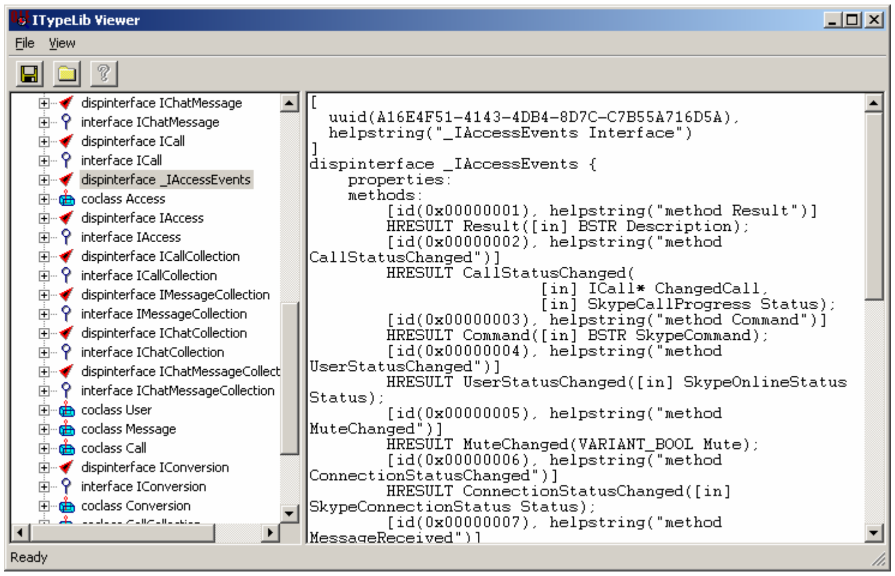
OLE/COM Object Viewer

```
SftTree/OCX Tree Control - Softel
Shacojin 1.0 Type Library (Ver 1.0)
shappmgr 1.0 Type Library (Ver 1.
Sheridan 3D Controls (Ver 1.0)
Sheridan 3D Controls (Ver 1.0)
Sheridan ActiveThreed Plus Contr
Shockwave ActiveX Control (Ver 1.
Shockwave Flash (Ver 1.0)
Signup 1.0 Type Library (Ver 1.0)
SkypeAPI 1.0 Type Library (Ver 1.
SkypePresence 1.0 Type Library (\
SlayerUI 1.0 Type Library (Ver 1.0
SlbCsp 1.0 Type Library (Ver 1.0)
Slider 1.0 Type Library (Ver 1.0)
SmartArgEditor 1.0 Type Library (\
Snapshot Viewer Control (Ver 1.0)
SoftietheSnowman 1.0 Type Librar
```

```
SkypeAPI 1.0 Type Library (Ver 1.0)
{1586AE5F-6BF1-4F8A-8199-0D5C745BF716}

TypeLib = OLE (Part 4 of 5)
  {1586AE5F-6BF1-4F8A-8199-0D5C745BF716}
    1.0 = SkypeAPI 1.0 Type Library
      0
        win32 = C:\Projects\SkypeAPI\ReleaseMinD
      FLAGS = 0
      HELPDIR = C:\Projects\SkypeAPI\ReleaseMinDe
```

Ready

Once you've selected the library, select the item **dispinterface _IAccessEvents**



ITypeLib Viewer

```
dispinterface IChatMessage
interface IChatMessage
dispinterface ICall
interface ICall
dispinterface _IAccessEvents
coclass Access
dispinterface IAccess
interface IAccess
dispinterface ICallCollection
interface ICallCollection
dispinterface IMessageCollection
interface IMessageCollection
dispinterface IChatCollection
interface IChatCollection
dispinterface IChatMessageCollect
interface IChatMessageCollection
coclass User
coclass Message
coclass Call
dispinterface IConversion
interface IConversion
coclass Conversion
```

```
[
  uuid(A16E4F51-4143-4DB4-8D7C-C7B55A716D5A),
  helpstring("_IAccessEvents Interface")
]
dispinterface _IAccessEvents {
    properties:
    methods:
        [id(0x00000001), helpstring("method Result")]
        HRESULT Result([in] BSTR Description);
        [id(0x00000002), helpstring("method
CallStatusChanged")]
        HRESULT CallStatusChanged(
                        [in] ICall* ChangedCall,
                        [in] SkypeCallProgress Status);
        [id(0x00000003), helpstring("method Command")]
        HRESULT Command([in] BSTR SkypeCommand);
        [id(0x00000004), helpstring("method
UserStatusChanged")]
        HRESULT UserStatusChanged([in] SkypeOnlineStatus
Status);
        [id(0x00000005), helpstring("method
MuteChanged")]
        HRESULT MuteChanged(VARIANT_BOOL Mute);
        [id(0x00000006), helpstring("method
ConnectionStatusChanged")]
        HRESULT ConnectionStatusChanged([in]
SkypeConnectionStatus Status);
        [id(0x00000007), helpstring("method
MessageReceived")]
```

Ready

There are two important pieces of information you'll need for each callback. The **id** and the **parameter list**. For example, you'll almost certainly need to handle the APIStatusChanged event. The prototype information is:

```
[id(0x00000009), helpstring("method
APIStatusChanged")]
        HRESULT
APIStatusChanged(SkypeAPIAttachmentStatus Status);
```

In your class, you'll need to define the following:

```
HRESULT __stdcall OnAPIStatusChanged (enum
SkypeAPIAttachmentStatus Status);
BEGIN_SINK_MAP(CMySkypeAppDlg)
            SINK_ENTRY_EX(IDC_MYSKYPE,
DIID__IAccessEvents, 0x9, OnAPIStatusChanged)
END_SINK_MAP()
```

Now, all of this would work, except there is bug with ATL when you implement IDispEventImpl and include callbacks with enumerated parameters. Basically, enumerated parameters get translated as VT_USERDEFINED, which won't work. For nasty details, you can read Microsoft's knowledge base article KB237771 (http://support.microsoft.com/default.aspx?scid=kb;en-us;237771). Fortunately, there is a pretty simple workaround. You have to override a function called GetFuncInfoFromId as follows:

```
virtual HRESULT GetFuncInfoFromId(const IID& iid,
DISPID dispidMember,
LCID lcid, _ATL_FUNC_INFO& info) {

// class base class implementation
HRESULT hr = IDispEventImpl<0x100, CSkypeTestCPPDlg,
&DIID__IAccessEvents, &LIBID_SKYPEAPILib, 1, 0>::
GetFuncInfoFromId(iid, dispidMember,lcid, info);
if (SUCCEEDED(hr)) {
if (::InlineIsEqualGUID(iid, DIID__IAccessEvents)) {
for(long l = 0; l < info.nParams; l++) {
if(info.pVarTypes[l] == VT_USERDEFINED) {
info.pVarTypes[l] = VT_I4;
}
}
}
      }
return hr;
}
```

CREATE OBJECTS

If you're not scared off by now, you're almost there. Now we just need to set up the functions to create our Skype API objects and clean them up when we're done. We'll define an Access object as well as a Conversion object within the ConnectToSkype function, and clean them up in the DisconnectFromSkype function. When you call these functions depends on the structure of your application. For an MFC dialog application, for example, the OnInitDialog handler is a good place to call ConnectToSkype, and the DestroyWindow handler is a good place to call DisconnectFromSkype. Here's an example of how to create the Skype objects and set up your class to receive event notifications:

```
void ConnectToSkype()
{
TCHAR msg[128];

// Create the Skype Access object and do event advise
HRESULT hr =
m_ptrAccess.CreateInstance(__uuidof(Access));
if(FAILED(hr)) {
_stprintf(msg, _T("Unable to create Skype Access
object: 0x%08X"), hr);
MessageBox(msg, "Error", MB_OK | MB_ICONSTOP);
} else {
IUnknownPtr pUnk;
m_ptrAccess.QueryInterface(IID_IUnknown, & pUnk);
hr = DispEventAdvise(pUnk);
pUnk.Release();
if(FAILED(hr)) {
_stprintf(msg, _T("Unable to capture events from
Skype Access object: 0x%08X"), hr);
MessageBox(msg, _T("Error"), MB_OK | MB_ICONSTOP);
}
}

// Create the Skype Conversion object so we can get text representation
of enum values
if(SUCCEEDED(hr)) {
hr =
m_ptrConversion.CreateInstance(__uuidof(Conversion));
if(FAILED(hr)) {
```

```
_stprintf(msg, _T("Unable to create Skype Conversion
utility object: 0x%08X"), hr);
MessageBox(msg, _T("Error"), MB_OK | MB_ICONSTOP);
}
}

if(SUCCEEDED(hr)) {
hr = m_ptrAccess->Connect();
if(FAILED(hr)) {
_stprintf(msg, _T("Unable to receive Skype events:
0x%08X"), hr);
MessageBox(msg, _T("Error"), MB_OK | MB_ICONSTOP);
}
}
}
```

Now, to clean up...

```
void DisconnectFromSkype()
{
if(m_ptrAccess) {
      IUnknownPtr pUnk;
      m_ptrAccess.QueryInterface(IID_IUnknown, &
pUnk);
      DispEventUnadvise(pUnk);
      pUnk.Release();
      m_ptrAccess.Release();
}

if(m_ptrConversion) {
      m_ptrConversion.Release();
}
}
```

Using the CSkypeMessageQueue Class

If you don't want to deal with all of the COM stuff, you may want to take a look at the CSkypeMessageQueue class in the Skype API source code. You can use it within your application to communicate with Skype, and get multithreading and Unicode/UTF-8 translation done for you. Examine the Access.cpp file to see how the CSkypeMessageQueue can be used.

Contributors:

Arie Bakker,
Netherlands
Skype Name:
`aria44`

Arie, contributor of the Simple Skype Answering Machine source code and executable, studied mathematics in the early 1970's, worked in the insurance industry, half time as mathematician, half time in automation (developing programs). For the past eight years Arie has done some programming for fun and sometimes as little projects for a few clients.

Kevin Delaney,
Ontario, Canada
Skype Name:
`sillyrabbit999`

Kevin, contributor of the sample Chat Message utility in Visual BASIC.NET, is a student by day and programmer by night. Since his first computer at age five, he's been hooked on exploring the computer and the Internet and learning how everything works. He has experience as a server administrator of several game servers and is currently the co-founder of the development team of www.khaoslabs.com. Kevin also designed and coded the new, soon-to-be-released QZoxy Presence System.

German Koninin,
Prague, Czech
Republic
Skype Name:

German, producer of SkypeTracer, is a Software Developer by occupation. His other interests are reverse engineering and sports.

Johnny Krogsgaard,
Copenhagen,
Denmark
Skype Name:
`john-9`

Johnny, developer of the first Visual BASIC 6 code for the Chat Message executable, works with an Internet-based company (www.andelsportal.dk) in Copenhagen where he's responsible for telephones, computers, server, homepage, etc. He has been programming since a very early age and is fluent in Java, PHP, HTML, ASP, Delphi, and Visual BASIC. He also has good skills in sense of graphical work with extensive experience with Photoshop, Fireworks and Flash. He has been a beta-tester for Skype since the start.

Wei Li, DSV/KTH, Sweden
Skype Name: `beichuang`

Wei, contributor of the commented Skype Java Programming using JSkype , is PhD student of Computer and Systems Sciences. He's doing research in "Ubiquitous and Context Aware Computing" and is currently involved in the Adaptive and Context-Aware System project (http://psi.verkstad.net/acas). www.dsv.su.se/fuse

Gaurav Prasad, India
Skype Name: `pagux2you`

Gaurav, contributor of the commented Visual BASIC 6 code and the sample Chat Message utility in Perl, is an Automation/ Process specialist. His core competencies are process optimization/redesign and rapid product development from business requirements to create business differentiation. www.pagux.com

Jason Terando, Huntington Beach, California, USA
Skype Name: `jasonterando`

Jason, contributor of the C++ code, is Manager of Application Development at Rapidtext, Inc., a firm specializing in transcription and captioning. He is the author of a COM wrapper for the Skype API available at KhaosLabs.com. He holds a Bachelor's degree in Management Information Systems from the California State University.

beeSync

The developer of the free SkypeX: www.beesync.com/skypex/index.html

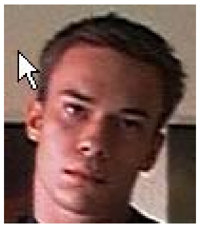
Translators:

Chinese:
Wuyijun,
Kunshan,Suzhou province,China
Skype Name:
`poseidem_wu`

Wuyijun is a 24-year-old software engineer at Compal, a manufacturer of computer notebooks. He will concentrate on compiling notebook BIOS source code this year. Because he has a natural curiosity to understand how things work, he will explore every detail of Skype and Skype-related topics. So it's time to study dotNet, not only Assembly Language.

Dutch:
The Netherlands
Skype Name:
`kootstra`

As an early adopter of Skype, `kootstra` has seen the online user base grow from a few thousand to well over a few million. Though his active forum days are now behind him, he's still an active Skype user and remains interested in the technology; though be it from the sidelines.



German:
Markus Daehne
Moers, NRW, Germany
Skype Name:
`uniquex`

Markus is a 24-year-old student of Applied Computer Science at the University in Duisburg. He's an administrator/moderator of the German Skype-Forum.com community and publisher of the meinSkype.de portal. He's currently focused on writing a book about Skype (to be published by Syngress) due to be available in autumn 2005. He's interested in all Skype-related subjects as well as programming and working with the Skype API.
www.meinskype.de

Hungarian and Romanian:
Peter Henning
Bucharest, Romania
Skype Name:
`madraven`

Peter is a 22-year-old student of Applied Computer Science and Economy at the Academy of Economic Studies in Bucharest. Resulting from his choice of university, he is knowledgeable in economics, software development (C++/C# and Assembler) and system administration, a combination he sees as a great advantage for his future career in product management and entrepreneurship. He's an active member of the Skype forum and a Skype for Windows closed Beta tester. He's interested in all Skype-related subjects as well as programming and working with the Skype API.

Italian:

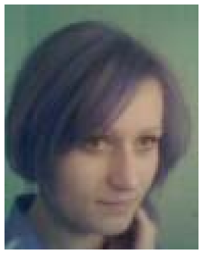
skaipe.com

Il Portale Italiano a Skype.
www.skaipe.com

Japanese:

Brian Harward & Seishi Isobe,
Nagoya, Japan

Brian has been living in Japan for the last 10 years. Five years ago, with the help of members from Rotary Naka Nagoya, he started @Works Ltd. They now run two Xserves and operate as a server service for the local community. The Japanese translation team includes Seishi Isobe. Seishi has been translating for @Works for the last three years and is a valued member of the @Works team. www.a-area.org



Polish:
Ewelina
Chudzińska,
Świdnica, Poland
Skype Name:
`linkaewe`

Ewelina is a 23-year-old student of English Philology, graduated from Central Metropolitan of TAFE, Perth, Western Australia. Currently working as a Polish-English interpreter.

Portuguese:
Paulo Sousa,
Póvoa de Varzim,
Portugal
Skype Name:
`pjrsousa`

Computer Science Graduate.
Born in Portugal
Post Graduate Studies: Electronic Commerce WebDesign; WebProgramming

Russian:
Nina Kupper,
Rudolstadt,
Germany
Skype Name:
`nina_tomsk`

Nina is 24-year-old freelance technical writer and translator of English/Russian. She is a graduate of the Tomsk (in Siberia) State University of Control Systems and Radio Electronics, possessing a System Technical Engineer degree.
Nina is married and lives in Germany. She’s interested in mastering the German language to, perhaps, start doing translation as well.

Spanish:
Ramon,
Madrid, Spain
Skype Name:
`quemasda`

Ramon is a Telecommunications Engineer. Currently he is working in a worldwide telecom company. He is the administrator and editor of the community of Hispanic Skype users, www.skype-es.com . He also likes programming and has experience with Skype API, especially by using .NET.

Brazilian Spanish: Tarcizio Pinto, Santa Mario, Rio Grande do Sul, Brazil

Skype Name: `tarciziorp`

Tarcizio is a Computer Science Academic at the UNIFRA University and an Electrical Engineer Academic at the UFSM University. He's a researcher of overheating in Athlon CPU's and Linux "distros" in old RISC PowerPC computers. In his spare time, Tracizio plays guitar with the Gorda Iliada band.

Testers:

A good programmer knows they can't test their own code. To manage quality requires a written test plan and qualified, committed testers. The story is no different when writing a book. So, I wrote a test plan and found a team of dedicated Skypers, from all parts of Skypeland, who represented the diverse audience of this guide to assess readability, comprehension and clarity of the internal procedures.

The guide has been tested with these classes of individuals—

1. English version with testers whose mother tongue is not English
2. Sophisticated programmers with Skype experience
3. Sophisticated programmers with no Skype experience
4. Novice programmers with Skype experience
5. No programming experience, but very Skype literate

Any errors or omissions fall squarely on my shoulders. A heartfelt thanks and, hopefully from you readers, loud applause go out to the following testers:

Cesar Andrade,
Piracicaba, SP, Brazil
Skype Name: `cesarandrade2005`

`ascenna` (Skype Name)

Arie Baker,
Baarn, Netherlands
Skype Name: `aria44`

Vir Banhu,
Knowledge Systems, Inc., Bangalore, India
Skype Name: `virbhanu`

Peter Henning,
Romania
Skype Name: `madraven`

`jaragrets` (Skype Name)

Neil Lindsey,
New Westminster, B.C. Canada
Skype Name: `neillindsey`

Jerry Pedersen,
Victoria, B.C. Canada
Skype Name: `jeryskype915`

Knowledge Systems, Inc.,
Bangalore, India
Skype Name: `sanith1976`

Rolf Stefani

Tomasz Tybulewicz,
Gdansk, Poland
Skype Name: `tomasztybulewicz`

Yann
China
Skype Name: `yannshen9100`

About the Author:

Bill Campbell

Kelowna, B.C., Canada

Skype Name: `theptcompany`

www.SkypeJournal.com

Bill's an avid user, enthusiast and advocate of Skype and has been since the product was six weeks old. He has over thirty years of management experience. Close to a third of that was spent in management positions with the Hewlett Packard Company.

He invented and managed the development of the first Skype Presence Server and Q-Card™ to broadcast Skype online status to the Skype Bulletin Board Forum and provide both status and voice interactivity for web sites, including the German Skype Community and Spanish Skype Community Forums.

Bill and Skype CEO, Niklas Zennström, co-signed a Non-Disclosure Agreement September 02, 2004 paving the way for his company QZoxy, Inc. to be invited as the first member of the *Skype Developer Program*.

He and his peers in the closed Skype Beta Forum help shape the future of Skype's products. They test all new Skype Products: Conference Calling, Multi-Chat, SkypeOut, SkypeIn, Video, API and Skype Workgroups. Bill is also a nurturer and advocate for new users on the Skype Support Forum and is one of the Top Ten posters.

As Technical Editor of the Skype Journal he works with his partner, Stuart Henshall, to keep members of the Skype community up-to-date on the latest information on Skype— news, commentary, events, interviews, product reviews and technical issues; especially anything to do with the Skype API and API Developers.

You can subscribe to be notified of new updates of the Skype Journal by copying this link – www.SkypeJournal.com/blog/atom.xml – and pasting it into your Rich Site Summary (RSS) reader and receive hourly or daily updates.