VoMS™ 1.0
(Voice Message Services)

User's Guide

Table of Contents

About this guide

This manual describes the features of the VoMS product, including how to install, configure and use the VoMS server and client components. The described features and how-to instructions apply to all supported platform, unless specifically noted in the text.

Intended audience

This document is for System Administrators and Software Developers.

Conventions used in this document

This section describes the style conventions used in this document.

Italic

An *italic* font is used for filenames, URLs, emphasized text, and the first usage of technical terms.

`Monospace`

A `monospaced` font is used for code fragments and data elements.

Bold

A **bold** font is used for important messages, names of options, names of controls and menu items, and keys.

User Input

Keys are rendered in **bold** to stand out from other text. Key combinations that are meant to be typed simultaneously are rendered with "+" sign between the keys, such as:

Ctrl+F

Keys that are meant to be typed in sequence will be separated with commas, for example:

Alt+S, H

This would mean that the user is expected to type the Alt and S keys simultaneously and then to type the H key.

Graphical symbols

This symbol is used to mark useful tips.

This symbol is used to indicate important notes.

Abbreviations and product reference terms

VoMS – Voice Message Service

Oracle – This refers to all supported Oracle® database servers

SQL Server – This refers to all versions of Microsoft® SQL Server™ database servers.

ASE – This refers to all versions of the Sybase® SQL Server™ and Sybase® Adaptive Server® Enterprise database servers.

ASA – This refers to all versions of the Sybase® Adaptive Server® Anywhere database servers.

DB2 – This refers to all versions of the IBM® DB2® database servers.

Trademarks

.

CHAPTER 1, How VoMS works

Message processing workflow

VoMS is designed specifically for distributed processing and client/server applications. The premise of such client/server computing is to delegate the execution of a voice messaging tasks to the server freeing client programs from the need to use the complicated telephony API and perform complex operations analyzing and converting input and output sound streams. The client is also freed from waiting for relatively slow phone dialing and voice message processing.

As a true client/server system VoMS allows multiple concurrent clients to connect to the VoMS server. Clients use one available VoMS application programming interface (API) to communicate with the server using standard TCP/IP protocol. Two APIs are available at this time: COM-based API and DLL-based API. These APIs allow virtually any Windows or ASP-based program to take advantage of the provided voice massaging services.

Components

Figure 1 shows the components of a message processing workflow diagram.

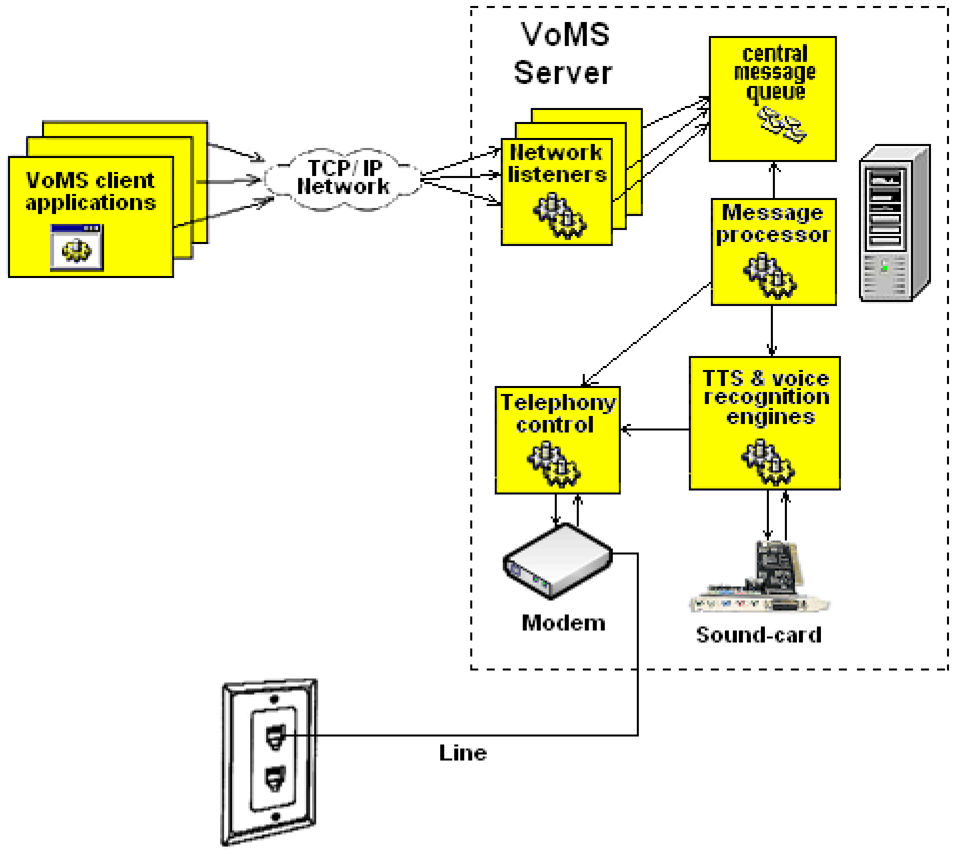


Figure 1: Message processing workflow components

The following paragraphs describe message workflow components.

VoMS client applications - Applications that use VoMS client API to send voice messages though VoMS server

Central message queue - Central message queue is the file based message queue where VoMS server stores submitted messages until they are processed.

Message processor - Message Processor is a special process responsible for processing queued messages and delivering messages to specified recipients. It is also responsible for reorganizing messages in the queue after initial message delivery attempts fail. Message Processor also performs message transformations from text-based and WAV based data submitted by clients to internal ready-to-send formants containing sound data only.

Telephony control – Telephony control is responsible for communicating with internal or external modems attached to the server and performing call progress detection with the help of the voice recognition engine.

TTS & voice recognition engines – TTS acronym stands for Text-To-Speech conversion. The Message Processor uses TTS engine to convert text-based message data to sound data segments and merge the resulting sound segments with other WAV segments. The voice recognition engine is used in call progress detection for recognizing line connection/disconnection events, as well as for human voice and answering machine detection.

Modem - Any standard voice-modem found in modern personal computers. Voice-modems are modems that are capable of playing and recording audio over a telephone line, in other words, they can send and receive WAV-encoded data. The modem is used to dial out and "record" and "play" WAV data.

Sound card - Any standard sound card found in modern personal computers. The sound-card is used by TTS engine to convert text based data into WAV format.

Workflow

The simple processing workflow can be described by following processing steps:

1. A custom application loads VoMS client component using the "create client " VoMS API method.

2. The application then calls "connect" VoMS client method to connect and logon to a local or remote VoMS server and establish a new client session.

3. The application creates message description in a plain text format and submits a new message to the VoMS server along with the telephone number of the message recipient. The VoMS client parses message description and any –pre-recorded sound files have been specified in the message, then client then transfers copies of these files to the VoMS server.

4. This step is optional and only required in situations when the client application must wait for the message processing to complete before continuing. If this is not required the application can skip step 4.

   The client application periodically calls ""get message status" VoMS client method to find out the current processing state of the specified message until the status becomes "completed" or "failed." Please note that the status becomes "failed" only after all attempts to rich the recipient failed. By default VoMS server will try to resend the message 3 more times with 5 minute breaks between attempts.

5. The application calls "disconnect" VoMS client method close the session, logoff and disconnect from the VoMS server.

6. The client application then calls "destroy" method to unload VoMS client components.

If the client application has a need to send multiple messages in one session it can repeat steps 3 and optionally 4 as many times as many messages it needs to send.

For specific information on how to call VoMS client methods see CHAPTER 2, VoMS Client API

Tip: VoMS server implements advanced message queuing which allows true high-performance asynchronous message processing, as the performance of the client applications is not affected by the slow performance of the voice call processing. VoMS client applications simply call client interface functions that quickly write messages to the message queue and immediately free applications making it unnecessary for the applications to wait for the slower voice call processing.

Supported voice messaging methods and formats

VoMS supports the following message types:

- Pre-recorded sound files in WAV format.

- Dynamically synthesized voice messages using advanced text-to-speech technologies. For dynamic text-to-speech messages VoMS server uses Microsoft Windows ® Speech API to generate dynamic sound files.

- Combination of pre-recorded sound files in WAV format and dynamically synthesized voice messages using advanced text-to-speech technologies. Each message can have unlimited number of pre-recorded and dynamically synthesized parts.

CHAPTER 2, Sending voice messages

Overview

The Voice Message Services (VoMS), is a TAPI-compliant system service, which allows users to make fully automated phone calls and send voice messages from their applications using either a local VoMS server or a shared network VoMS server. The VoMS can perform the following business functions:

- Send pre-recorded messages in standard WAV file format
- Send dynamically synthesized voice messages using text-to-speech technology
- Send mixed messages that contain multiple parts including pre-recorded and dynamically synthesized segments.

The VoMS server makes required phone calls, verifies message format and files, assembles multiple message parts into single messages, performs call progress monitoring, detects call answer and then plays the messages.

Advanced options

VoMS server supports a number of options that can be used to customize voice message delivery. The following options can be configured using VoMS Administrator Console:

- Modem selection and configuration
- Default computer voice for dynamically synthesized voice messages, including voice name, rate and volume
- Call progress detection parameters, timeouts, number of retries.
- VoMS server security and users
- Other message queuing and processing options

For more information on VoMS software configuration and settings see Configuring VoMS Server topic.

Message sending methods

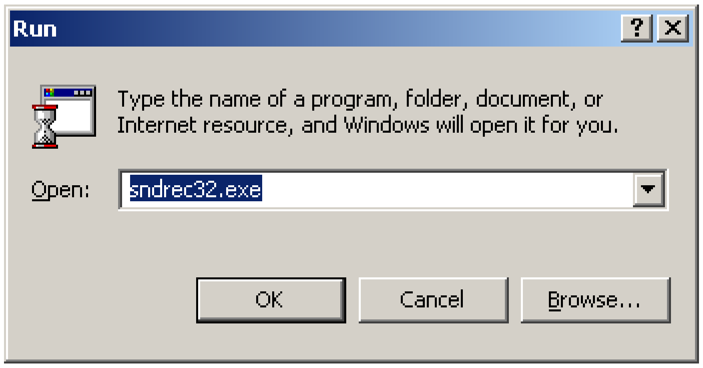
You can send voice messages using any of the following methods:

- Custom VoMS client application calling VoMS client API functions used to communicate with the VoMS server. For more information on this method read CHAPTER 3, VoMS Client API.
- VoMS command line client utility run with parameters from DOS command line. This utility can be also run from batch files and from Windows applications. For more information on this method read CHAPTER 4, VoMS Command Line Interface.

Creating pre-recorded sound messages and message segments

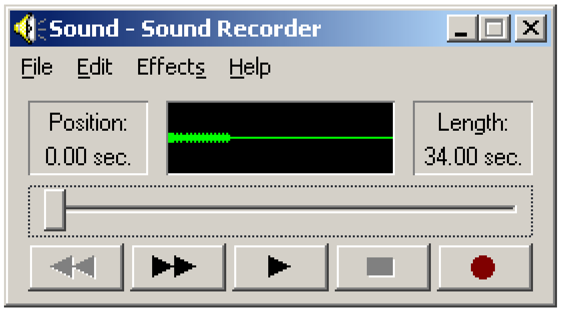
Use standard Windows Sound Recorder utility or similar software to create pre-recorded messages and save them as .WAV file. To record sounds you must have a microphone and sound card installed on your computer.

A shortcut to the standard Windows Sound Recorder utility can be normally found in **C:\Documents and Settings\All Users\Start Menu\Programs\Accessories\Entertainment** folder. The utility can be also started using **Run** option in the Windows Start menu. The name of this program is sndrec32.exe.


Run
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.
Open:
OK
Cancel
Browse...

Open: sndrec32.exe

The following screenshot demonstrates the Sound Recorder utility graphical interface.



For detailed instruction on how to use this utility click Help/Contents menu available in the Sound Recorder graphical interface. After you are done with the sound recording copy all created .WAV files to the directory accessible from your database server. If your database server is running on a Unix or other non-Windows system you can use FTP or other appropriate methods to copy WAV files.

The recorded WAV files can be then attached to voice messages.

For detailed descriptions of how to send sound files and specify their position in multiple-part messages see description and usage examples in the following chapters.

Creating dynamically synthesized voice messages and message segments

VoMS software features built-in methods for dynamically synthesized voice messages using text-to-speech technology provided with most Windows installations. On your part, you simply call one of the

VoMS APIs or use the command line utility and specify text that you want to the computer to speak to the message recipient using computer synthesized human voice. VoMS software automatically takes care of the rest. It automatically converts text messages to sound files, dials specified phone numbers, analyzes call recipient responses are "speaks" your messages in case if a successfully human voice or answering machine response is detected.

CHAPTER 3, VoMS Client API

The Voice Message Services Client (VoMS Client) application programming interface (API) allows you to incorporate basic voice messaging functionality for users in client applications. This technology is available on computers that are running Microsoft Windows 98 and later.

The VoMS Client API for includes 2 distinct API implementations:

- Implementation of Win32 DLL using standard calling conventions common in Win32 APIs.
- Implementation of COM dual interfaces.

The COM API is intended for use by developers and system administrators who are familiar with Visual Basic, VBScript, JavaScript or other scripting languages supporting Microsoft Windows Component Object Model (COM).

The DLL API is intended for third-party vendors who are familiar with C++ or other programming language supporting dynamic library linking.

Tip: The COM API internally is build as a wrapper to the DLL API

Overview of VoMS client API Methods

Instantiating VoMS client

An instance of VoMS client object must be created before you can call VoMS client functions.

After you instantiate a VoMS client object you can use it to connect to a VoMS server, send messages and check on the status of sent messages.

Connecting to VoMS server

In most situations, a VoMS client application must establish a connection to a VoMS server before the application can use other voice messaging features.

To establish the connection to the VoMS server, a VoMS client application must call the **OpenSession** function successfully before it sends any message.

Querying connection status

A VoMS client application can check its connection state by calling **IsConnected** method.

To establish the connection to the VoMS server, a VoMS client application must call the **OpenSession** function. To close a previously opened session it must call **CloseSession** function.

Sending messages

Call **SendMessage** function to send voice message description to the VoMS server and transmit all referenced sound files.

Obtaining last message ID

Every message submitted to a VoMS server is assigned unique message number also called Message ID. VoMS client applications can use this unique number to check message processing status at any time. Call **GetLastMessageID** function immediately after successful execution of the **SendMessage** function to obtain message ID of the last submitted message. The obtained message id can be then used with the **GetMessageStatus** function to check message processing status.

Querying client processing status

A VoMS client application can query its processing state by calling **GetStatus** method. This method can be called at any time even before a VoMS server session has been opened.

The following statuses can be reported:

- Message is currently queued for processing."
- Message is being sent right now."
- At least one attempt to send this message failed. Next attempt to send it will be made in a few minutes."
- All attempts to send this message have failed. Modem or phone line is not ready."
- Invalid message data. Message could not be sent."
- All attempts to send this message have failed. Number is busy."
- All attempts to send this message have failed. No answer."
- All attempts to send this message have failed. Bad circuit or invalid phone number."
- Could not send message. Invalid line answer. Timeout waiting for silence."
- Message has been partially sent. Hung up detected before message completion."
- Message has been sent after detection of human voice response."
- Message has been sent after detection of answering machine response or mailbox greeting."
- Server was unable to find this message or report on the message status. Probably message archival options are not configured."
- Server was unable to find this message or report on the message status.
- Last message ID is nnn. [Current message processing status reported by the VoMS server]
- You are not connected to the server.

Querying message processing status

A VoMS client application can query processing status of the last message by calling **GetMessageStatus** method. This method can be called only while an active VoMS server session is opened.

The following statuses can be returned:

- 1000 -- Message is still queued.
- 1001 -- Message is being sent.
- 1002 -- Message is waiting for redial after failure.
- 1003 -- Message failed. Bad message format.
- 1004 -- Message failed. Message data invalid or cannot be loaded/processed
- 1005 -- Message failed. All attempts to reach the destination phone number failed, line is busy.
- 1006 -- Message failed. Message not played, no line pickup detected.
- 1007 -- Message failed. Bad circuit, invalid phone number.
- 1008 -- Message failed. Greeting taking too long.
- 1009 -- Message playback aborted. Hung up detected during playback.
- 1010 – Human voice detected. Message successfully sent.

- 1011 – Answering machine detected. Message successfully sent.
- 1012 -- Message not found. Message archiving disabled or invalid message ID.

Validating WAV file consistency

A VoMS client application can check consistency of a WAV file by calling **ValidateWavFile** method. This method can be called at any time even before a VoMS server session is opened.

Disconnecting from VoMS server

It is the responsibility of the calling VoMS client application to disconnect from the VoMS server.

A VoMS client application must call the **CloseSession** function as the last function before it terminates. The function disconnects the calling application from the VoMS server and deallocates all previously allocated resources on the server and client sides.

COM Interface

About COM Interface

In programming, the Component Object Model (COM), also known as ActiveX, is a Microsoft technology for software componentry. It is used to enable cross-software communication. Although it has been implemented on several platforms, it is primarily used with Microsoft Windows. Its precursor was object linking and embedding (OLE)

Each COM component has a unique identity, which exposes interfaces that allow applications and other components to access their features. COM components are more versatile than Win32 DLLs because they are completely language-independent, have built-in inter-process communications capability, and easily fit into an object-oriented program design.

Using COM API in Your Program

The following steps summarize typical programming tasks required to incorporate the functionality of the VoMS API in a client application if you are using the COM implementation.

To code a client application

1. Create an instance of a COM object by calling the **CreateObject** function (or similar function available in your programming language). Then connect to an active VoMS server by calling the **OpenSession** method of the COM object. For more information, see OpenSession method description.
2. After creating an instance of a COM object for a specific server call necessary API methods for performing the required tasks.
3. Terminate the connection to the VoMS server by calling the **CloseSession** method of the COM object. For more information, see CloseSession topic.

A Quick Example: Visual Basic application

The following paragraph provides a quick example of calling VoMS client COM methods from Visual Basic code.

1. Declare COM object variable

```
Dim Client As Object
Dim RC As Boolean
```

2. Create VoMS client COM objects, using its programmatic identifier (ProgID="VoMSClientCOM.VoMSClient") and check that the creation was successful:

```
Set Client = CreateObject("VoMSClientCOM.VoMSClient")

If Client Is Nothing Then
  ' Handle the error ...
  MsgBox "Error creating COM object..."
End If
```

3. Access methods of the VoMS client using automation syntax:

```
' Open new session
RC = Client.OpenSession("MyServer", 1020, "John Doe", False)
If RC = False Then
    ' Handle the error
    MsgBox Client.GetLastError()
Else
    ' Send simple dynamically generated message
    RC = Client.SendMessage("Hello world", "123-456-7890", 0)
    ' Done. Close session
    RC = Client.CloseSession()
End If

' Destroy COM object and release all allocated resources
Set Client = Nothing
```

Instantiating VoMS client

An instance of VoMS client COM object must be created before you can call VoMS client functions.

If you are writing a C/C++ application, you must call the CoCreateInstance function to retrieve a pointer to a **_VoMSClient** interface and create an instance of a **VoMSClient** object.

If you are writing a Visual Basic application, you must call the Visual Basic **CreateObject** function to create an instance of a **VoMSClient** object.

If you are writing an application in some other programming language refer to your programming language documentation for information on how to create a COM object

After you instantiate a VoMS client object you can use to connect to a VoMS server

Examples:

1. Visual Basic example

```
Dim Client As Object
On Error Resume Next
Set Client = CreateObject("VoMSClientCOM.VoMSClient")
If Client = Nothing Then MsgBox "Error creating VoMS object"
```

2. C/C++ example

```
// Initialize COM interface
HRESULT hr = CoInitialize(NULL);
if (FAILED(hr))
{
   printf("CoInitialize failed\n");
   return;
}

// Instantiate VoMS client object, obtaining interface pointer
_VoMSClient* pClient;
hr = CoCreateInstance(CLSID_VoMSClient, NULL, CLSCTX_SERVER,
                       IID__VoMSClient, (void **) & pClient);
if (FAILED(hr))
{
   printf("CoCreateInstance failed\n");
   return;
}
```

3. PowerBuilder example

```
integer result
OLEObject Client

Client = CREATE OLEObject
result = Client.ConnectToNewObject("VoMSClientCOM.VoMSClient")
if result <> 0 then MessageBox("Error", "Error creating VoMS object")
```

4. Delphi example

```
Var
   Client: Variant;
begin
   try
```

```
            Client := CreateOleObject('VoMSClientCOM.VoMSClient');
         except
            Application.MessageBox('Error creating VoMS object',
                                    'Error', MB_OK);
         end;
      end;
```

Destroying VoMS client

A previously created instance of VoMS client COM object must be destroy after you are done with the VoMS processing.

If you are writing a C/C++ application, you must call the Release function to release a pointer to a _**VoMSClient** interface and allow previously created instance of **VoMSClient** object to deallocate itself. You may also need to call CoUninitialize function to clean up the remaining COM interface resources.

If you are writing a Visual Basic application, you must assign the **VoMSClient** object variable to **Nothing** using the **Set** statement in order to destroy that instance and deallocate all associated resources.

If you are writing an application in some other programming language refer to your programming language documentation for information on how to destroy a COM object

Tip: If a VoMS server session has not been closed before distraction the **VoMSClient** object will automatically disconnect from the server and close the session

Examples:

1. Visual Basic example

```
    ' Create VoMS client
    Dim Client As Object
    Set Client = CreateObject("VoMSClientCOM.VoMSClient")

    ' Now destroy VoMS client
    Client = Nothing
```

2. C/C++ example

```
    // Initialize COM interface
    CoInitialize(NULL);

    // Instantiate VoMS client object, obtaining interface pointer
    _VoMSClient* pClient;
    CoCreateInstance(CLSID_VoMSClient, NULL, CLSCTX_SERVER,
                           IID__VoMSClient, (void **) & pClient);

    // Now release object reference
    pClient.Release();
    // Clean up COM resources
    CoUninitialize();
```

3. PowerBuilder example

```
// Create VoMS client
OLEObject Client
Client = CREATE OLEObject
Client.ConnectToNewObject("VoMSClientCOM.VoMSClient")

// Now destroy VoMS client
Client.DisconnectObject()
DESTROY Client
```

4. Delphi example

```
Var
   Client: Variant;
begin
   // Create VoMS client
   Client := CreateOleObject('VoMSClient');

   // Now destroy VoMS client
   Client := Unassigned;
end;
```

CloseSession

Boolean CloseSession( )

The **CloseSession** function closes active VoMS client session, logoffs and disconnects user from a VoMS server.

Return: Returns TRUE if it succeeds or FALSE if an error occurs. Use the GetLastError function to obtain the error message.

Parameters: None

See also:
OpenSession
IsConnected

Examples:

1. Visual Basic example

```
' Create VoMS Client
Dim Client As Object
Set Client = CreateObject("VoMSClientCOM.VoMSClient")

' Open new session
Client.OpenSession "MyServer", 1020, "John Doe", False

' Send simple dynamically generated message
Client.SendMessage "Hello world", "123-456-7890", 0

' Done. Close session
Client.CloseSession
Set Client = Nothing
```

GetLastMessageID

Long GetLastMessageID( )

The **GetLastMessageID** function reports unique message ID of the last voice message submitted from the current VoMS client session.

Return: Returns unique message ID or zero if no messages have been sent from the current session.

Parameters: None

See also:

OpenSession
SendMessage
GetMessageStatus

Examples:

1. Visual Basic example

```
' Create VoMS Client
Dim MessageID As Long
Dim Client As Object
Set Client = CreateObject("VoMSClientCOM.VoMSClient")

' Open new session
Client.OpenSession "MyServer", 1020, "John Doe", False

' Send simple dynamically generated message
Client.SendMessage "Hello world", "123-456-7890", 0

' Obtain ID of the sent message
MessageID = Client.GetLastMessageID()
MsgBox "Last message ID is " & CStr(MessageID)

' Done. Close session
Client.CloseSession
Set Client = Nothing
```

GetLastError

String GetLastError( )

The **GetLastError** function reports text of the last error.

Return: Returns error text or an empty string no errors occurred in the current VoMS client instance.

Parameters: None

See also:

- OpenSession
- SendMessage
- GetStatus
- GetMessageStatus

Examples:

1. Visual Basic example

```
' Create VoMS Client
Dim Client As Object
Dim Ok As Boolean
Set Client = CreateObject("VoMSClientCOM.VoMSClient")

' Open new session
Ok = Client.OpenSession("MyServer", 1020, "John Doe", False)
If Not Ok Then
    ' Handle the error
    MsgBox Client.GetLastError()
Else
    ' Send simple dynamically generated message
    Ok = Client.SendMessage("Hello world", "123-456-7890", 0)
    If Not Ok Then
        ' Handle the error
        MsgBox Client.GetLastError()
    End If
    ' Done. Close session
    Client.CloseSession()
End If

' Destroy COM object and release all allocated resources
Set Client = Nothing
```

GetMessageStatus

Long GetMessageStatus(Long MessageID)

The **GetMessageStatus** function retrieves processing state for the specified queued, active or already processed voice message at the specified VoMS server.

Return: Returns message status. See Querying message processing status topic for the complete list of supported status values and their descriptions.

Parameters:

MessageID	Valid message number identifying a queued or active or already processed voice message.

See also:

OpenSession
SendMessage
GetStatus

Examples:

1. Visual Basic example

```
' Declare Win API function
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

' Create VoMS Client
Dim Client As Object
Dim Status As Long
Dim MessageID As Long
Set Client = CreateObject("VoMSClientCOM.VoMSClient")

' Open new session
Client.OpenSession "MyServer", 1020, "John Doe", False

' Send simple dynamically generated message
Client.SendMessage "Hello world", "123-456-7890", 0
' Obtain ID of the sent message
MessageID  = Client.GetLastMessageID()


' Wait for the message processing completion
Status = Client.GetMessageStatus(MessageID )
Loop While Status < 103
   Sleep(5000)
   Status = Client.GetMessageStatus(MessageID )
Wend

' Done. Close session
Client.CloseSession
' Destroy COM object and release all allocated resources
Set Client = Nothing
```

GetStatus

String GetStatus( )

The **GetStatus** reports VoMS client current state in plain text format. This function can be used in graphical interactive and non-interactive applications to report on the progress of work and also in other areas where status messages are displayed.

Return: Returns description of the current processing status. See Querying client processing status topic for the complete list of supported status messages.

Parameters: None

See also:

OpenSession
SendMessage
GetMessageStatus
GetLastError

Examples:

1. Visual Basic example

```
' Create VoMS Client
Dim Client As Object
Set Client = CreateObject("VoMSClientCOM.VoMSClient")

' Open new session and get connection status
Client.OpenSession "MyServer", 1020, "John Doe", False
MsgBox "Current status: " & Client.GetStatus()

' Send simple dynamically generated message and then
' get message processing status
Client.SendMessage "Hello world", "123-456-7890", 0
MsgBox "Current status: " & Client.GetStatus()
' ... do something else here and get new status...
MsgBox "Current status: " & Client.GetStatus()

' Done. Close session and get connection status
Client.CloseSession
MsgBox "Current status: " & Client.GetStatus()
' Destroy COM object and release all allocated resources
Set Client = Nothing
```

IsConnected

Boolean IsConnected( )

The **IsConnected** function reports VoMS client connectivity status. This function can be called at any time.

Return: Returns TRUE if a VoMS session is currently active and connected to a VoMS server returns or FALSE otherwise.

Parameters: None

See also:
OpenSession
CloseSession

Examples:

2. Visual Basic example

```
' Create VoMS Client
Dim Client As Object
Set Client = CreateObject("VoMSClientCOM.VoMSClient")

' Test connectivity state
If Not Client.IsConnected() Then MsgBox "Not connected"

' Open new session
Client.OpenSession "MyServer", 1020, "John Doe", False

' Test connectivity state
If Client.IsConnected() Then MsgBox "Connected"

' Done. Close session
Client.CloseSession
Set Client = Nothing
```

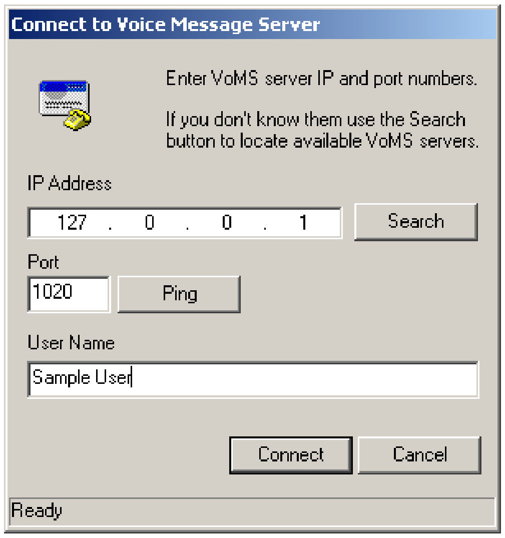
OpenSession

Boolean OpenSession(String ServerLocation,
Long ServerPort,
String UserName,
Boolean ShowConnectDialog)

The **OpenSession** function connects to VoMS server, performs user logon and if successful starts a new VoMS client session.

Return: Returns TRUE if it succeeds or FALSE if an error occurs. Use the GetLastError function to obtain the error message.

Parameters:

ServerLocation	Computer network name or IP address of the VoMS server. If an empty string or LocalHost is specified the client will attempt to connect to a locally running VoMS server.
ServerPort	Port number used by the VoMS server for network connections. The default port number is 1020.
UserName	Name of the user who is establishing new session. If empty string is specified VoMS client will use name of the user who is currently logged on to the computer running VoMS client application.
ShowConnectDialog	This parameter controls whether VoMS client should display the **Connect To Voice Message Server** dialog. If the parameter value is TRUE the dialog is displayed on the screen and values of all other parameters are used as default values displayed on the dialog. Do not set this parameter to TRUE in applications that run unattended. Connect to Voice Message Server Enter VoMS server IP and port numbers. If you don't know them use the Search button to locate available VoMS servers. IP Address: 127 . 0 . 0 . 1 Port: 1020 User Name: Sample User Ready

See also:
CloseSession
IsConnected

Examples:

1. Visual Basic example

```
' Create VoMS Client
Dim Client As Object
Set Client = CreateObject("VoMSClientCOM.VoMSClient")

' Open new session
Client.OpenSession "MyServer", 1020, "John Doe", False

' Send simple dynamically generated message
Client.SendMessage "Hello world", "123-456-7890", 0

' Done. Close session
Client.CloseSession
Set Client = Nothing
```

SendMessage

Boolean SendMessage(String MessageDescriptor,
String Telephone,
Long Priority)

A VoMS client application calls the **SendMessageSession** function to place a new voice message to the VoMS server message queue.

Return: Returns TRUE if it succeeds or FALSE if an error occurs. Use the GetLastError function to obtain the error message.

Parameters:

MessageDescriptor	Message description containing optional text you want the computer to speak to the phone call recipient and also containing optional pre-recorded sound files. Use pair of **<wav>**and **</wav>** tags to include names of pre-recorded sound files. Example descriptors *Say <wav>c:\voices\hello.wav</wav> to my friend* This message descriptor instructs VoMS server to perform the following processing steps: 1. Convert word "Say" and the trailing space character into a WAV file using computer synthesized human voice. 2. Append pre-recorded hello.wav file to the WAV file created in step 1. 3. Convert words "*to my friend* " and the leading space character into a WAV file using computer synthesized human voice. 4. Append last WAV file created in step 3 to the file created in step 1 and updated in step 2. Here is another example that uses 2 pre-recorded sound files and 1 text segment as a computer synthesized text-to-speech insertion. *<wav>reminder.wav</wav> 10:00 AM <wav>call_to_cancel.wav</vaw>* This message descriptor instructs VoMS server to perform the following processing steps: 1. Convert text "10:00 AM " including leading and trailing spaces into a WAV file using computer synthesized human voice. 2. Concatenate *reminder.wav* file with the WAV created in step 2 and then with *call_to_cancel.wav* file.
Telephone	Message recipient telephone number. Specify the number exactly as you would dial it from your phone including any dial out numbers, long distance codes, local area codes and so on. Digits in the number can be separated by optional dashes, parenthesis, spaces and other non-digit symbols, for example, 1 (212) 555-6789. **Tip**: In addition to the digits 0 through 9, many modems support the following special characters to control the dialing sequence: • A comma "," character (without the quotes) – will cause your mode to pause for about 2 or 3 seconds. • A capital letter "W" (without the quotes) - to wait for a dial tone. • An "@" symbol (without the quotes) - to wait for silence, before dialing.

	• A explanation mark "!" (without the quotes) - to cause a hook flash.
Priority	Message processing priority .Message processing priority takes a value of **0**, **1**, or **2**. Messages having higher priority numbers are inserted into the message queue before messages having lower priority numbers.

See also:

OpenSession
GetLastMessageID
GetMessageStatus

Examples:

1. Visual Basic example

```
Dim Client As Object
Dim Price As Float, Phone As String, Message As String
Price = 10.54
Phone = "123-456-7890"

' Create VoMS Client
Set Client = CreateObject("VoMSClientCOM.VoMSClient")

' Open new session
Client.OpenSession "MyServer", 1020, "StockAlerter", False

' Submit message to the message queue
Message = "<wav>c:\sound\stock_alert.wav</wav>  " & _
          Format(Price, "#,##0.00") & "  dollars  "_
          "<wav>c:\sound\call_to_trade.wav</wav>"
Client.SendMessage Message, Phone, 0

' Done. Close session
Client.CloseSession
Set Client = Nothing
```

ValidateWavFile

Boolean ValidateWavFile(String FileName)

The **ValidateWavFile** function checks consistency of a WAV file. This function can be called at any time even before a VoMS server session is opened.

Return: Returns TRUE if the function did not find any problems within the file data; or returns FALSE otherwise. The FALSE result indicates that the file cannot be played.

Parameters:

FileName	Full name of a WAV file including path.

See also:

SendMessage

Examples:

1. Visual Basic example

```
' Create VoMS Client
Dim Client As Object
Set Client = CreateObject("VoMSClientCOM.VoMSClient")

' ' Check file
If Not Client.ValidateWavFile( "c:\voices\hello.wav" ) Then
    ' Handle the error
    MsgBox "File c:\voices\hello.wav is corrupted and cannot be send"
End If


' Destroy COM object and release all allocated resources
Set Client = Nothing
```

DLL Interface

About DLL Interface

The VoMS client DLL interface consists of C functions implemented in dynamically linked library **VoMSClient.dll**. The interface implemented in the same manner as Windows 32-bit API using __stdcall calling conventions. Any application or programming system that supports calling DLL functions can use the VoMS client DLL interface.

Using DLL API in Your Program

The following steps summarize typical programming tasks required to incorporate the functionality of the VoMS API in a client application if you are using the DLL implementation.

To code a client application

1. Declare function prototypes describing VoMS client function names and parameters.
2. If the programming system does not automatically handle DLL loading and unloading call the necessary method to load VoMSClient.dll file and resolve addresses of declared functions within the DLL module. For example in C/C++ you can use LoadLibrary and GetProcAddress functions for this purpose.
3. Instantiate VoMS client interface using VOMS_CreateClient function then connect to an active VoMS server by calling the VOMS_Connect function.
4. After successfully connection call necessary API methods for performing the required tasks.
5. Terminate the connection to the VoMS server by calling the VOMS_Disconnect function and then destroy VoMS client instance using VOMS_DestroyClient function.
6. If the programming system does not automatically handle DLL loading and unloading call the necessary method to unload VoMSClient.dll file. For example in C/C++ you can use FreeLibrary function for this purpose.

A Quick Example: Visual Basic application

The following paragraph provides a quick example of calling VoMS client COM methods from Visual Basic code.

1. Declare function prototypes

```
Option Explicit

Declare Function VOMS_CreateClient Lib "VoMSClient.dll" () As Long

Declare Sub VOMS_DestroyClient Lib "VoMSClient.dll" _
   (ByVal OBJPTR As Long)

Declare Function VOMS_Connect Lib "VoMSClient.dll" _
   (ByVal OBJPTR As Long, ByVal ShowOptions As Boolean, _
   ByVal IPAddress As String, ByVal Port As Long, _
   ByVal UserName As String) As Long

Declare Function VOMS_Disconnect Lib "VoMSClient.dll" _
```

```
   (ByVal OBJPTR As Long) As Long

Declare Function VOMS_IsConnected Lib "VoMSClient.dll" _
   (ByVal OBJPTR As Long) As Boolean

Declare Function VOMS_SendMessage Lib "VoMSClient.dll" _
   (ByVal OBJPTR As Long, ByVal Message As String, _
   ByVal Phone As String, ByVal Priority As Long) As Long

Declare Function VOMS_ CheckWavFileFormat Lib "VoMSClient.dll" _
   (ByVal OBJPTR As Long, ByVal FileName As String) As Boolean

Declare Function VOMS_GetStatus Lib "VoMSClient.dll" _
   (ByVal OBJPTR As Long,  ByVal MessageID As Long) As Long

Declare Function VOMS_GetLastError Lib "VoMSClient.dll" _
   (ByVal OBJPTR As Long, ByVal ErrorBuffer As String, _
   ByVal BufferSize As Long) As Long
```

2. Declare variables

```
Dim ClientPtr As Long
Dim RC As Boolean
```

3. Create VoMS client instance

```
ClientPtr = VOMS_CreateClient()

If ClientPtr = 0 Then
  ' Handle the error ...
  MsgBox "Error creating VoMS client instance..."
End If
```

4. Access methods of the VoMS client using declared functions:

```
' Open new session and connect to the server
RC = VOMS_Connect(ClientPtr, True, "MyServer", 1020, "John Doe")
If RC = False Then
    ' Handle the error
    MsgBox VOMS_GetLastError(ClientPtr)
Else
    ' Send simple dynamically generated message
    RC = VOMS_SendMessage(ClientPTR, "Hello world", "123-456-7890", 0)
    ' Done. Close session and disconnect from the server
    RC =VOMS_Disconnect(ClientPtr)
End If

' Destroy VoMS client and release all allocated resources
VOMS_Destroy(ClientPtr)
```

Instantiating VoMS client

An instance of VoMS client must be created before you can call VoMS client functions. Use **VOMS_CreateClient** to create new instance of VoMS client. After you instantiate a VoMS client object you can use to connect to a VoMS server.

LPVOID VOMS_CreateClient(VOID);

Return: Returns pointer to the VoMS client instance if it succeeds or NULL if an error occurs.

Parameters: None

See also:
VOMS_Connect
Destroying VoMS client

Examples:

1. Visual Basic example

```
Declare Function VOMS_CreateClient Lib "VoMSClient.dll" () As Long

Dim ClientPtr As Long
ClientPtr = VOMS_CreateClient()
If ClientPtr = 0 Then MsgBox "Error creating VoMS client"
```

2. C/C++ example

```
typedef VOID* (LPFNDLLFUNC)(VOID);

HINSTANCE hDLL;               // Handle to DLL
LPFNDLLFUNC lpfnDllFunc;      // Function pointer
LPVOID lpClient = NULL;        // VoMS client pointer

// Load VoMSClient.dll dynamic-link library
hDLL = LoadLibrary("VoMSClient.dll");
if (hDLL != NULL)
{
   // Get address of VOMS_CreateClient function
   lpfnDllFunc = (LPFNDLLFUNC)GetProcAddress(hDLL, "VOMS_CreateClient");
   if (!lpfnDllFunc)
   {
      // handle the error
      FreeLibrary(hDLL);
      return SOME_ERROR_CODE;
   }
   else
   {
      // call the function
      lpClient = lpfnDllFunc();
      if (!lpClient)
      {
         // handle the error
         FreeLibrary(hDLL);
         return SOME_ERROR_CODE;
      }
   }
}
```

3. PowerBuilder example

```
Function Long VOMS_CreateClient() Library "VoMSClient.dll"

long ClientPtr

ClientPtr = VOMS_CreateClient()
if ClientPtr = 0 then MessageBox("Error", "Error creating VoMS client")
```

4. Delphi example

```
function VOMS_CreateClient: Integer; cdecl; external 'VoMSClient.dll';

var
   ClientPtr: Integer;
begin
   ClientPtr = VOMS_CreateClient;
   if ClientPtr = 0 then
      MessageBox('Error creating VoMS client', 'Error', MB_OK);
end;
```

Destroying VoMS client

A previously created instance of VoMS client must be destroy after you are done with the VoMS processing.

Use **VOMS_DestroyClient** to destroy an existing instance of VoMS client. If the client is connected to a VoMS server you must first disconnect from the server. Failure to close the connection may lead to resource leaking and other unforeseen consequences.

VOID VOMS_DestroyClient (LPVOID lpClient);

Return: None

Parameters:

lpClient	Pointer to VoMS client instance returned by VOMS_CreateClient function.

See also:
VOMS_Disconnect
Instantiating VoMS client

Examples:

1. Visual Basic example

```
Declare Sub VOMS_DestroyClient Lib "VoMSClient.dll" _
                                        (ByVal Client As Long)

Dim ClientPtr As Long
'...Create VoMS client...

' ...Call other functions...

' Destroy VoMS client and deallocate resources
VOMS_CreateClient ClientPtr
```

2. C/C++ example

```
typedef VOID (LPFNDLLFUNC)(VOID *);

HINSTANCE hDLL;               // Handle to DLL
LPFNDLLFUNC lpfnDllFunc;      // Function pointer
VOID* lpClient = NULL;        // VoMS client pointer

// ...Load VoMSClient.dll dynamic-link library...

// ...Instantiate VoMS client...

// ...Call other functions...

// Get address of VOMS_DestroyClient function
   lpfnDllFunc = (LPFNDLLFUNC)GetProcAddress(hDLL,"VOMS_DestroyClient");
   if (!lpfnDllFunc)
      // handle the error
      return SOME_ERROR_CODE;
   else
      // call the function
      lpfnDllFunc(lpClient);
}
```

3. PowerBuilder example

```
Subroutine VOMS_DestroyClient(long pClient) Library "VoMSClient.dll"

long ClientPtr

// ...Create VoMS client...

// ...Call other functions...

// Destroy VoMS client and deallocate resources
VOMS_DestroyClient(ClientPtr)
```

4. Delphi example

```
procedure VOMS_DestroyClient(Client: Integer); cdecl; external
   'VoMSClient.dll';

var
   ClientPtr: Integer;
begin
   // ...Create VoMS client...

   // ...Call other functions...

   // Destroy VoMS client and deallocate resources
   VOMS_DestroyClient(ClientPtr);
end;
```

VOMS_CheckWavFileFormat

BOOL VOMS_CheckWavFileFormat(LPVOID lpClient, LPCSTR szFileName)

The **VOMS_CheckWavFileFormat** function checks consistency of a WAV file. This function can be called at any time even before a VoMS server session is opened.

Return: Returns TRUE if the function did not find any problems within the file data; or returns FALSE otherwise. The FALSE result indicates that the file cannot be played.

Parameters:

lpClient	Pointer to a VoMS client instance. **lpClient** parameter must refer to a valid VoMS client instance created using VOMS_CreateClient function.
szFileName	Pointer to a null-terminated string whose value is the full name of a WAV file including path.

See also:
    VOMS_SendMessage

Examples:

1. Visual Basic example

```
' Declare external functions
Declare Function VOMS_CreateClient Lib "VoMSClient.dll" () As Long
Declare Function VOMS_ CheckWavFileFormat Lib "VoMSClient.dll" _
   (ByVal OBJPTR As Long, ByVal FileName As String) As Boolean

' Create VoMS Client
Dim ClientPtr As Long
ClientPtr = VOMS_CreateClient


' Check hello.wav file
If Not VOMS_ CheckWavFileFormat(ClientPtr, "c:\greetings\hello.wav") Then
   MsgBox "File c:\greetings\hello.wav is invalid and cannot be sent"
End If
```

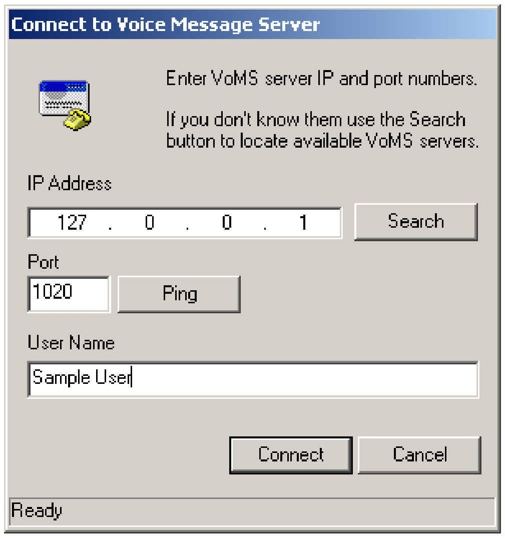
VOMS_Connect

BOOL VOMS_Connect(LPVOID lpClient, BOOL bShowConnectDialog, LPCSTR szServerLocation, DWORD dwServerPort, LPCSTR szUserName)

The **VOMS_Connect** function connects to VoMS server, performs user logon and if successful starts a new VoMS client session. VoMS client instance must be created using VOMS_CreateClient function prior to calling **VOMS_Connect**.

Return: Returns TRUE if it succeeds or FALSE if an error occurs. Use the VOMS_GetLastError function to obtain the error message.

Parameters:

lpClient	Pointer to a VoMS client instance. **lpClient** parameter must refer to a valid VoMS client instance created using VOMS_CreateClient function.
bShowConnectDialog	This parameter controls whether VoMS client should display the **Connect To Voice Message Server** dialog. If the parameter value is TRUE the dialog is displayed on the screen and values of all other parameters are used as default values displayed on the dialog. Do not set this parameter to TRUE in applications that run unattended.  Connect to Voice Message Server Enter VoMS server IP and port numbers. If you don't know them use the Search button to locate available VoMS servers. IP Address: 127 . 0 . 0 . 1 Port: 1020 User Name: Sample User Ready
szServerLocation	Pointer to a null-terminated string whose value is computer network name or IP address of the VoMS server. If a NULL string or LocalHost is specified the client will attempt to connect to a locally running VoMS server.
dwServerPort	Port number used by the VoMS server for network connections. The default port number is 1020.
szUserName	Pointer to a null-terminated string whose value is name of the user starting the new session. If NULL string is specified, VoMS client will use name of the user who is currently logged on to the computer running VoMS client application.

See also:

Instantiating VoMS client
VOMS_IsConnected

Examples:

1. Visual Basic example

```
' Declare external functions
Declare Function VOMS_CreateClient Lib "VoMSClient.dll" () As Long
Declare Function VOMS_Connect Lib "VoMSClient.dll" _
   (ByVal OBJPTR As Long, ByVal ShowOptions As Boolean, _
   ByVal IPAddress As String, ByVal Port As Long, _
   ByVal UserName As String) As Long

' Create VoMS Client
Dim ClientPtr As Long
ClientPtr = VOMS_CreateClient

' Open new session
VOMS_Connect ClientPtr, False, "MyServer", 1020, "John Doe"
```

VOMS_Disconnect

VOID VOMS_Disconnect(LPVOID lpClient)

The **VOMS_Disconnect** function closes active VoMS client session, logoffs and disconnects user from a VoMS server.

Tip: **VOMS_Disconnect** function does not release all resources allocated to the VoMS Client instance. To release call these resources call VOMS_DestroyClient function.

Return: Returns TRUE if it succeeds or FALSE if an error occurs. Use the VOMS_GetLastError function to obtain the error message.

Parameters:

lpClient	Pointer to VoMS client instance returned by VOMS_CreateClient function.

See also:
OpenSession
IsConnected

Examples:

1. Visual Basic example

```
' Declare external functions
Declare Function VOMS_CreateClient Lib "VoMSClient.dll" () As Long
Declare Function VOMS_Connect Lib "VoMSClient.dll" _
   (ByVal OBJPTR As Long, ByVal ShowOptions As Boolean, _
   ByVal IPAddress As String, ByVal Port As Long, _
   ByVal UserName As String) As Long
Declare Function VOMS_Disconnect Lib "VoMSClient.dll" _
   (ByVal OBJPTR As Long) As Long
Declare Sub VOMS_DestroyClient Lib "VoMSClient.dll" _
   (ByVal OBJPTR As Long)

' Create VoMS Client
Dim ClientPtr As Long
ClientPtr = VOMS_CreateClient


' Open new session
VOMS_Connect ClientPtr, False, "MyServer", 1020, "John Doe"

' Close session and destroy client instance
VOMS_Disconnect ClientPtr
VOMS_DestroyClient ClientPtr
```

VOMS_GetLastError

DWORD VOMS_GetLastError(LPVOID lpClient, LPSTR szBuffer, DWORD dwBufferSize)

The **VOMS_GetLastError** function reports text of the last error for the specified VoMS client instance. The null-terminated error text is copied to the specified output buffer **szBuffer**. If the size of the buffer is insufficient the copied text is cut to **dwBufferSize** characters.

Return: If **dwBufferSize** parameter is not zero the function returns number of characters copied to the output buffer; otherwise it returns the length of the error text not including the trailing null-character.

Parameters:

lpClient	Pointer to VoMS client instance returned by VOMS_CreateClient function.
szBuffer	Pointer to a buffer that receives the null-terminated string whose values is the description of the last error.
dwBufferSize	This parameter specifies the size of the output buffer **szBuffer**

Usage: To find out size of the required output buffer specify zero for the **dwBufferSize** parameter. Allocate the required memory chunk using the returned value plus 1 for the trailing null character and then call **VOMS_GetLastError** function again this time specifying real buffer size for the **dwBufferSize** parameter.

The error text is guaranteed not to exceed 1024 characters. If you don't want to call **VOMS_GetLastError** function twice you pre-allocate 1024 bytes for the output buffer and call **VOMS_GetLastError** function only once.

See also:
VOMS_Connect
VOMS_SendMessage
VOMS_GetStatus
VOMS_Disconnect

Examples:

2. Visual Basic example

```
' Declare external functions
Declare Function VOMS_CreateClient Lib "VoMSClient.dll" () As Long
Declare Function VOMS_Connect Lib "VoMSClient.dll" _
   (ByVal OBJPTR As Long, ByVal ShowOptions As Boolean, _
   ByVal IPAddress As String, ByVal Port As Long, _
   ByVal UserName As String) As Long
Declare Function VOMS_GetLastError Lib "VoMSClient.dll" _
   (ByVal OBJPTR As Long, ByVal ErrorBuffer As String, _
   ByVal BufferSize As Long) As Long

' Create VoMS Client
Dim ClientPtr As Long
Dim Ok As Boolean, ErrorBuffer As String
ClientPtr = VOMS_CreateClient
```

```
' Open new session
Ok = VOMS_Connect(ClientPtr, False, "MyServer", 1020, "John Doe")
If Not Ok Then
    ' Handle the error
    ErrorBuffer = String(1024, vbNullChar)
    VOMS_GetLastError ClientPtr, ErrorBuffer, 1024
    MsgBox ErrorBuffer
Else
    ' Send simple dynamically generated message
    Ok = Client.SendMessage("Hello world", "123-456-7890", 0)
    If Not Ok Then
        ' Handle the error
        ErrorBuffer = String(1024, vbNullChar)
        VOMS_GetLastError ClientPtr, ErrorBuffer, 1024
        MsgBox ErrorBuffer
    End If
End If
```

VOMS_GetStatus

DWORD VOMS_GetStatus (LPVOID lpClient, INT iMessageID)

The **VOMS_GetStatus** function retrieves processing state for the specified queued, active or already processed voice message at the specified VoMS server.

Return: Returns message status. See Querying message processing status topic for the complete list of supported status values and their descriptions.

Parameters:

lpClient	Pointer to VoMS client instance returned by VOMS_CreateClient function.
iMessageID	Valid message number identifying a queued or active or already processed voice message.

See also:
VOMS_SendMessage

Examples:

1. Visual Basic example

```
' Declare Win API function
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
' Declare external functions
Declare Function VOMS_CreateClient Lib "VoMSClient.dll" () As Long
Declare Function VOMS_Connect Lib "VoMSClient.dll" _
    (ByVal OBJPTR As Long,ByVal ShowOptions As Boolean, _
    ByVal IPAddress As String, ByVal Port As Long, _
    ByVal UserName As String) As Long
Declare Function VOMS_SendMessage Lib "VoMSClient.dll" _
    (ByVal OBJPTR As Long, ByVal Message As String, _
    ByVal Phone As String, ByVal Priority As Long) As Long
Declare Function VOMS_GetStatus Lib "VoMSClient.dll" _
    (ByVal OBJPTR As Long,  ByVal MessageID As Long) As Long

' Create VoMS Client
Dim ClientPtr As Long
ClientPtr = VOMS_CreateClient

' Open new session
VOMS_Connect ClientPtr, False, "MyServer", 1020, "John Doe"

' Send simple dynamically generated message
MessageID = VOMS_SendMessage(ClientPtr, "Hello world", "123-456-7890", 0)

' Wait for the message processing completion
Status = VOMS_GetStatus(ClientPtr, MessageID)
Loop While Status < 103
    Sleep(5000)
    Status = VOMS_GetStatus(ClientPtr, MessageID)
Wend

' Done. Continue processing ...
```

VOMS_IsConnected

BOOL VOMS_IsConnected(LPVOID lpClient)

The VOMS_**IsConnected** function reports VoMS client connectivity status.

Return: Returns TRUE if a VoMS session is currently active and connected to a VoMS server returns or FALSE otherwise.

Parameters:

lpClient	Pointer to VoMS client instance returned by VOMS_CreateClient function.

See also:

VOMS_Connect
VOMS_Disconnect

Examples:

1. Visual Basic example

```
' Declare external functions
Declare Function VOMS_CreateClient Lib "VoMSClient.dll" () As Long
Declare Function VOMS_Connect Lib "VoMSClient.dll" _
    (ByVal OBJPTR As Long, ByVal ShowOptions As Boolean, _
    ByVal IPAddress As String, ByVal Port As Long, _
    ByVal UserName As String) As Long
Declare Function VOMS_IsConnected Lib "VoMSClient.dll" _
    (ByVal OBJPTR As Long) As Boolean

' Create VoMS Client
Dim ClientPtr As Long
ClientPtr = VOMS_CreateClient

' Test connectivity state
If Not VOMS_IsConnected(ClientPtr) Then MsgBox "Not connected"

' Open new session
VOMS_Connect ClientPtr, False, "MyServer", 1020, "John Doe"

' Test connectivity state
If VOMS_IsConnected(ClientPtr) Then MsgBox "Connected"
```

VOMS_SendMessage

Boolean SendMessage(String MessageDescriptor,
String Telephone,
Long Priority)

A VoMS client application calls the **SendMessageSession** function to place a new voice message to the VoMS server message queue.

Return: Returns TRUE if it succeeds or FALSE if an error occurs. Use the GetLastError function to obtain the error message.

Parameters:

MessageDescriptor	Message description containing optional text you want the computer to speak to the phone call recipient and also containing optional pre-recorded sound files. Use pair of **<wav>**and **</wav>** tags to include names of pre-recorded sound files. Example descriptors *Say <wav>c:\voices\hello.wav</wav> to my friend* This message descriptor instructs VoMS server to perform the following processing steps: 5. Convert word "Say" and the trailing space character into a WAV file using computer synthesized human voice. 6. Append pre-recorded hello.wav file to the WAV file created in step 1. 7. Convert words "*to my friend* " and the leading space character into a WAV file using computer synthesized human voice. 8. Append last WAV file created in step 3 to the file created in step 1 and updated in step 2. Here is another example that uses 2 pre-recorded sound files and 1 text segment as a computer synthesized text-to-speech insertion. *<wav>reminder.wav</wav> 10:00 AM <wav>call_to_cancel.wav</vaw>* This message descriptor instructs VoMS server to perform the following processing steps: 9. Convert text "10:00 AM " including leading and trailing spaces into a WAV file using computer synthesized human voice. 10. Concatenate *reminder.wav* file with the WAV created in step 2 and then with *call_to_cancel.wav* file.
Telephone	Message recipient telephone number. Specify the number exactly as you would dial it from your phone including any dial out numbers, long distance codes, local area codes and so on. Digits in the number can be separated by optional dashes, parenthesis, spaces and other non-digit symbols, for example, 1 (212) 555-6789. **Tip**: In addition to the digits 0 through 9, many modems support the following special characters to control the dialing sequence: • A comma "," character (without the quotes) – will cause your mode to pause for about 2 or 3 seconds. • A capital letter "W" (without the quotes) - to wait for a dial tone. • An "@" symbol (without the quotes) - to wait for silence, before dialing.

	• A explanation mark "!" (without the quotes) - to cause a hook flash.
Priority	Message processing priority .Message processing priority takes a value of **0**, **1**, or **2**. Messages having higher priority numbers are inserted into the message queue before messages having lower priority numbers.

See also:

OpenSession
GetLastMessageID
GetMessageStatus

Examples:

2. Visual Basic example

```
Dim Client As Object
Dim Price As Float, Phone As String, Message As String
Price = 10.54
Phone = "123-456-7890"

' Create VoMS Client
Set Client = CreateObject("VoMSClientCOM.VoMSClient")

' Open new session
Client.OpenSession "MyServer", 1020, "StockAlerter", False

' Submit message to the message queue
Message = "<wav>c:\sound\stock_alert.wav</wav>  " & _
          Format(Price, "#,##0.00") & "  dollars  "_
          "<wav>c:\sound\call_to_trade.wav</wav>"
Client.SendMessage Message, Phone, 0

' Done. Close session
Client.CloseSession
Set Client = Nothing
```

CHAPTER 4, VoMS Command Line Interface

VoMS client software contains VMSEND.EXE command utility that can be used to submit voice messages for processing on local and remote VoMS servers.

Command format and supported command line parameters:

```
VMSend [/?] [/B] [/S server] [/P port] [/U user] [/X priority] /T telephone
       [/F file-to-play] [/M message-to-play]
```

Parameter description:

/?	Print help screen.
/B	Suppress banner printing.
/S	VoMS server computer IP or network name. If not specified, the default LocalHost is used.
/P	VoMS server port number. If not specified, the default 1020 port is used.
/U	Name of the user sending voice message. If not specified, **vmsend** uses the name of the user currently logged to the computer.
/X	Message processing priority .Message processing priority takes a value of **0**, **1**, or **2**. Messages having higher priority numbers are inserted into the message queue before messages having lower priority numbers. If not specified value 1indicating normal priority is used.
/T	Phone number where to send the message.
/F	Name of the text file containing message data. The file may contain both text-to-speech strings and <VAW> tags for inclusion of pre- recorded WAV files.
/M	Message data. This could be a simple text for a text-to speech message or a complex message containing both text and <VAW> tags for inclusion of pre- recorded WAV files.

Important notes:

- Either /F or /M must be specified in command parameters.
- Any parameter containing spaces must be enclosed in double quotes.

Examples:

1. This example command sends simple "Hello world" message using dynamic text-to-speech

synthesis and a local VoMS server.

```
vmsend /T 123-245-3344 /M "Hello world"
```

2. This example command sends pre-recorded "Hello world" message using remote VoMS server. The actual sound is recorded in *C:\My Messages\Hello world.wav* file.

```
vmsend /S "192.168.0.50" /T 123-245-3344 /M "<wav>C:\My Messages\Hello
world.wav</wav>"
```

3. This example command sends combination of 4 pre-recorded parts and 2 dynamically synthesized message parts.

```
vmsend /T 123-245-3344 /M "<wav>Hello.wav</wav>
<wav>Appoitment.wav</wav> 10:00 a.m. <wav>Number.wav</wav> 142
<wav>Goodbye.wav</wav>"
```

CHAPTER 5, Sample Programs and Code Examples

In addition to many code snippets available in this manual in CHAPTER 3, a number of ready-to-run examples can be installed with the VoMS client software. To install these examples, make sure the Examples option is checked in the installer.

The following examples demonstrate how easy it is to use VoMS client API in various computing environments.

Visual Basic

Visual Basic example application files are installed in *[VoMS home]\Examples\Visual Basic* directory. These files demonstrate how to use VoMS **DLL API**.

To run this example application you would need Microsoft Visual Basic 5 or later.

Delphi

Delphi example application files are installed in *[VoMS home]\Examples\Delphi* directory. These files demonstrate how to use VoMS **COM API**.

To run this example application you would need Borland Delphi 6 or later.

C/C++

C/C++ example application files are installed in *[VoMS home]\Examples\CPP* directory. These files demonstrate how to use VoMS **DLL API**.

To run this example application you would need Microsoft Visual Studio 6 or later.

PowerBuilder

C/C++ example application files are installed in *[VoMS home]\Examples\PowerBuilder* directory. These files demonstrate how to use VoMS **COM API**.

To run this example application you would need Sybase PowerBuilder 8 or later.

VBScript

VBScript example application files are installed in *[VoMS home]\Examples\VBScript* directory. These files demonstrate how to use VoMS **COM API**.

To run this example application you would need Windows Scripting Host and VBScript 2 or later installed on the system.

JavaScript

JavaScript example application files are installed in *[VoMS home]\Examples\JavaScript* directory. These files demonstrate how to use VoMS **COM API**.

To run this example application you would need Windows Scripting Host and JScript 2 or later installed on the system.

CHAPTER 6, Installation and Uninstallation

This topic describes what you need to install in order to use Voice Messaging Services (VoMS).

Installing VoMS is a relatively simple task, provided the system requirements are met. We will look at some of the details of performing the installation in this topic.

Server Installation

The computer where you are installing VoMS Server must feature the following components:

1. Windows 98 or later.
2. Network card with TCP protocol support enabled.
3. Internal or external voice modem supporting voice functions (most modern modems support voice-messaging functions; however you must check your modem documentation to ensure the required compatibility). Alternatively, the computer can feature Intel Dialog telephony board.
4. Sound-card compatible with the Microsoft Speech API.

Installation steps

The VoMS setup program provides a straightforward interface for VoMS installation.

When prompted what you want to install choose Voice Messaging Services **Server** option and then simply follow the Setup Wizard that will guide you through the entire installation process. When installing on a Windows NT environment, you should be logged on using an account that is a member of the local Administrators group before you install the program.

Please note the following points with regard to the installation:

- Keep the License Key supplied by Soft Tree Technologies handy. This will need to be entered during the installation.
- If you are installing VoMS on a trial basis, leave the Voice Message Server License Key field blank when prompted. This will provide you a temporary license for a 30-day trial period. The License can then be purchased from SoftTree Technologies and entered in later.
- By default, the Setup will place VoMS server shortcut into the Startup folder.
- At the end of the software installation to disk, you will be prompted to start the VoMS server and use the VoMS Administrator utility to configure modem connection settings and other voice messaging properties. If you chose not to configure Voice Messaging Services at this time, you can invoke the ‘Administrative Console’ shortcut from the Voice Message Services program group at a later time.

Important notes:
By default VoMS server is installed without a password meaning that anyone on your network who has VoMS Administrator utility installed cannot connect to the VoMS server. It is highly recommended that immediately after the installation you use the VoMS Administrator utility to set a password for all administrative connections. If you choose to set a password, click the **Password** button available on the **Users** screen. Enter new password and press the OK button to save it. The password will be stored in encrypted format in the system registry and all consecutive connections

to the VoMS server will require the same password entered on the VoMS Administrator connection dialog.

Client Installation

The computer where you are installing VoMS Server must feature the following components:

1. Windows 95 or later.
2. Network card with TCP protocol support enabled.

Installation steps

The VoMS setup program provides a straightforward interface for VoMS installation.

When prompted what you want to install choose Voice Messaging Services **Client** option and then simply follow the Setup Wizard that will guide you through the entire installation process. When installing on a Windows NT environment, you should be logged on using an account that is a member of the local Administrators group before you install the program.

Please note the following points with regard to the installation:

- Keep the License Key supplied by Soft Tree Technologies handy. This will need to be entered during the installation.
- If you are installing VoMS on a trial basis, leave the Voice Message Client License Key field blank when prompted. This will provide you a temporary license for a 30-day trial period. The License can then be purchased from SoftTree Technologies and entered in later.

Important notes: VoMS server modem options must be configured before you can use the first installed VoMS client.

Uninstallation

Both VoMS Server and Client supports standard uninstallation mechanism for removing program files from the system where VoMS was previously installed.

To uninstall VoMS do the following:

1. Click Windows **Start** button, from the Start Menu select **Settings**, then **Control Panel**.
2. Double-click **Add/Remove Programs**.
3. Select the **Voice Message Services** item in the programs list, click **Add/Remove** button

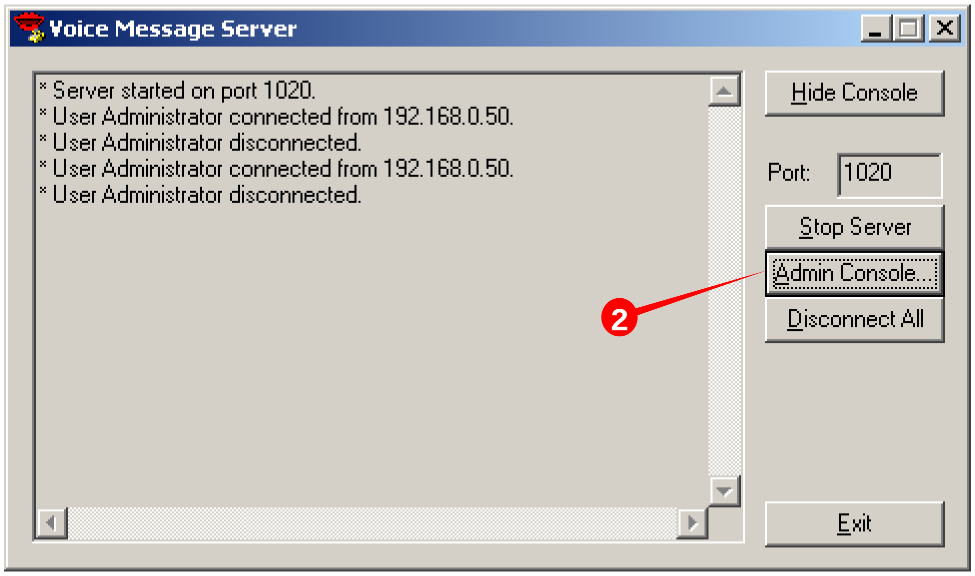
Delete all files that are left behind in the VoMS server home directory. Delete the home directory.

Configuring VoMS Server

After successful VoMS server installation you need to configure modem and voice message processing options. For information on how to install VoMS software see VoMS Installation topic.

Perform the following steps to configure your VoMS server:

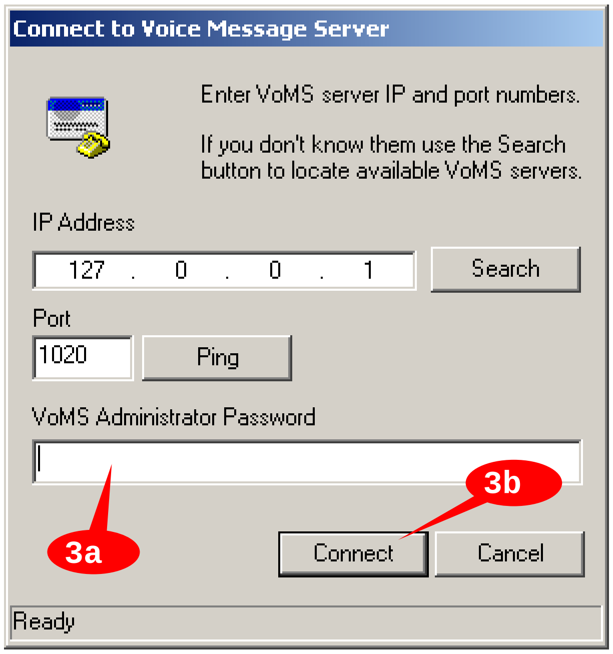
1. If VoMS server is not running you can launch it using VoMS Server shortcut from the Voice Message Server program group. If the server is already running double-click on the VoMS Server icon displayed in the system tray. This will display the **Voice Message Server** screen as on the following screenshot.



```
* Server started on port 1020.
* User Administrator connected from 192.168.0.50.
* User Administrator disconnected.
* User Administrator connected from 192.168.0.50.
* User Administrator disconnected.
```

Port: 1020

2. Click the **Admin Console** button. This will start the VoMS Administrator Console which will first display the **Connect To Voice Message Server** dialog.


Connect to Voice Message Server

Enter VoMS server IP and port numbers.

If you don't know them use the Search button to locate available VoMS servers.

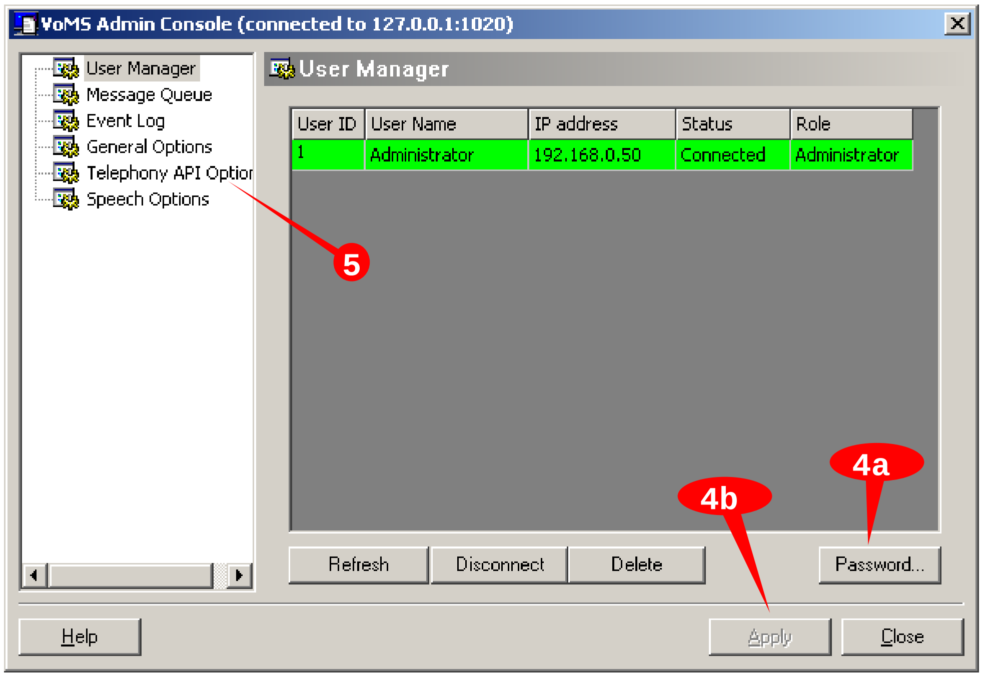
IP Address: 127 . 0 . 0 . 1

Port: 1020

VoMS Administrator Password

Ready

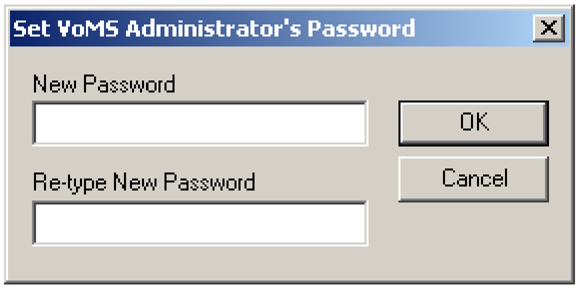
3. If you are connection first time and no password has been set yet skip this part; otherwise type your VoMS Administrator password into the **Password** field. Click the **Connect** button. If the connection is successful the **VoMS Admin Console** screen will appear.


VoMS Admin Console (connected to 127.0.0.1:1020)

User Manager

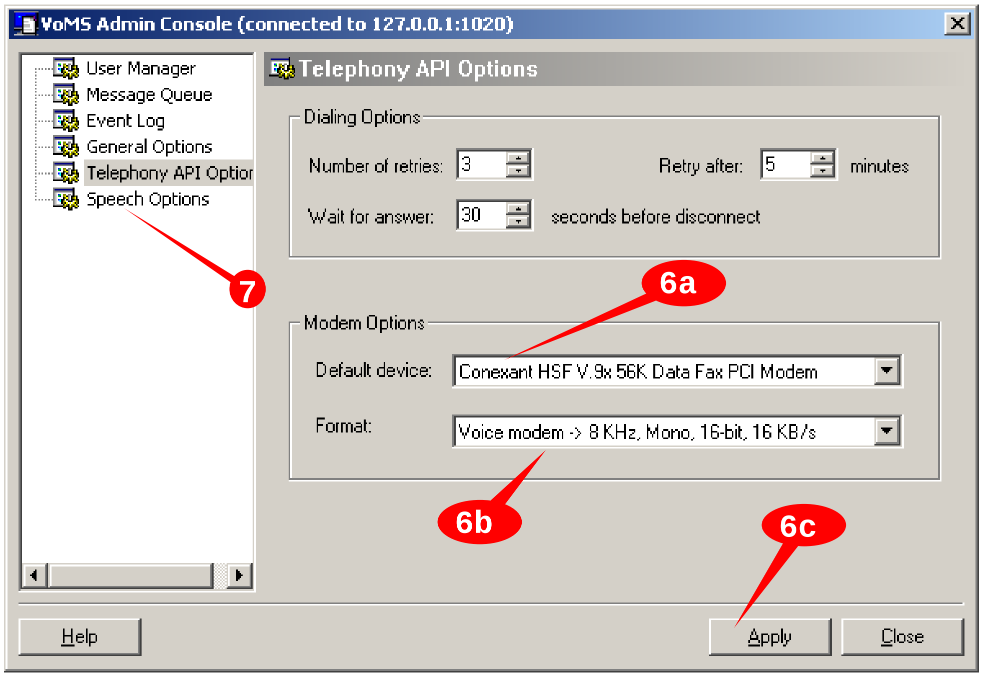
User ID	User Name	IP address	Status	Role
1	Administrator	192.168.0.50	Connected	Administrator

4. Click the **Password** button to set or update VoMS server password. This password will be required for all future connections. The **Set Password** dialog will appear.



Enter the new password, retype it in the second field and click the OK button to close the dialog. Make sure to click the **Apply** button to save changes.

5. Click the **Telephony API Options** item displayed on the left hand side of the VoMS Admin Console screen.



Dialing Options

Number of retries: 3

Retry after: 5 minutes

Wait for answer: 30 seconds before disconnect

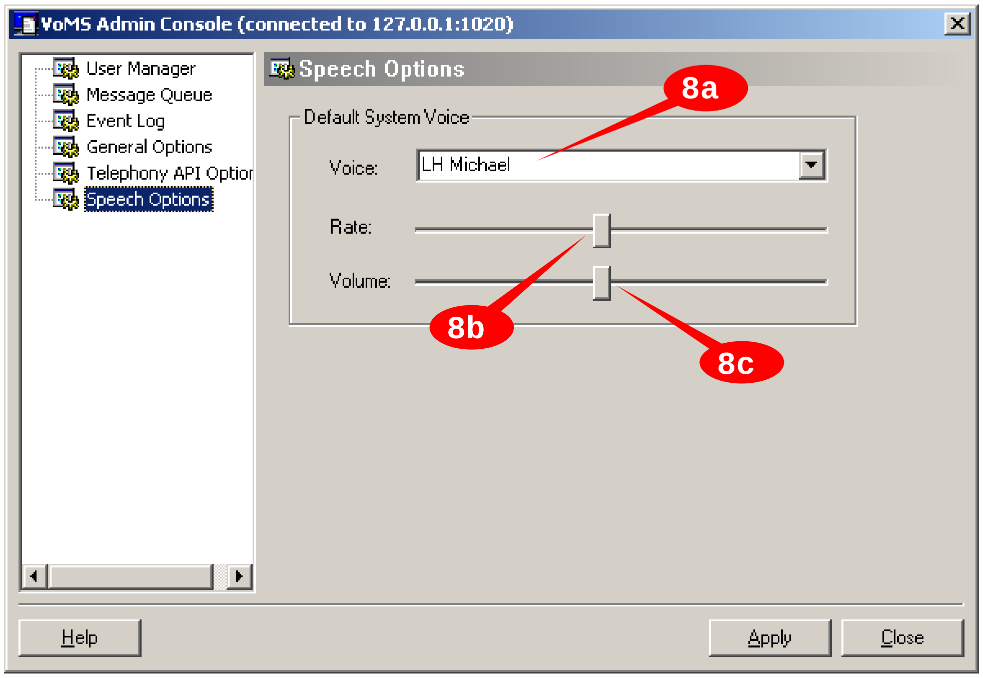
Modem Options

Default device: Conexant HSF V.9x 56K Data Fax PCI Modem

Format: Voice modem -> 8 KHz, Mono, 16-bit, 16 KB/s

6. In the **Default Device** drop-down box select name of the modem you want to use. In case if you have selected an Intel Dialog board change value in the **Format** drop-down list to Dialog board ->11.025 KHz. Click the **Apply** button to save changes.

7. Click the **Speech Options** button displayed on the left hand side of the VoMS Admin Console screen.



8. In the **Voice** drop-down box select name of the default system voice you want to use for computer synthesized text-to-speech messages. Using slide-bar controls select **Rate** and **Volume** values that you normally use for pre-recorded sound messages. Click the **Apply** button to save changes.

   For information on how to create pre-recorded messages see Creating pre-recorded sound messages topic in *CHAPTER 2*.

9. All other options such as log locations, logging levels, queue allocation and message archiving are not really important for the possessing. You can configure them as you see them fit your requirements. When you are done with all changes click the **Close** button to close the VoMS Admin Console.

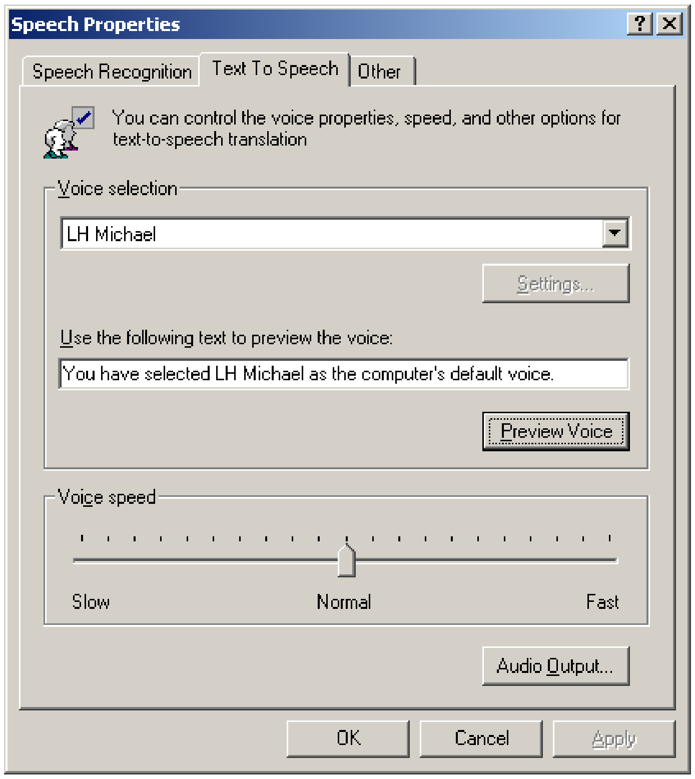
CHAPTER 7, Helpful Tips and Recommendations

Use the following helpful performance and usage tips when sending messages. These tips can greatly improve VoMS efficiency.

- For broadcasting large pre-recorded WAV files to lots of recipients submit such messages locally from the VoMS server computer. This way you can avoid transferring of lots of large files over your network.

- Experiment with different voices for Text-To-Speech messaging. Different voices work differently on different systems. Some voices could produce more clear and natural sounds then other. To try different voices without sending any test messages, open up Windows Control Panel and then double-click the **Speech** icon.

  Activate **Text-To-Speech** tab page. Use Voice selection box to select different voices and play test messages.

  You can control the voice properties, speed, and other options for text-to-speech translation

  Voice selection: LH Michael

  Use the following text to preview the voice: You have selected LH Michael as the computer's default voice.

  After you find the best voice, run **VoMS Administrator** utility and set the VoMS server to use this voice.

- You cannot verify statuses of sent or failed messages if the message archiving option is disabled. Always keep this option enabled unless free disk space is an issue on your server.

- You can use free FileDir utility to periodically zip or delete old archived files. Schedule this utility to run once a month or once a week or as often as you want.

- **VoMS Administrator** utility is installed by default on the VoMS server computer only. However this utility can be used to manage remote VoMS servers. If your license permits install VoMS server software on your workstation and from there run the **VoMS Administrator** utility.

CHAPTER 8, Troubleshooting and Maintenance

Basic Troubleshooting

You can perform some basic troubleshooting by checking the various directories and files. For example, setting the logging level in VoMS server settings to **High** and then checking the VOMS.LOG file in the 'C:\Program Files\Voice Message Services' directory. Usually, you should be able to figure out if a syntax or semantic mistake has been made. Check the APPENDIX C, Hardware and Software Requirements for further details.

Troubleshooting message processing

To effectively troubleshoot message processing anomalies, set the logging-level in the VoMS server settings to **Debug**. Reproduce the situation in which you believe messages are not processed correctly and then check the VOMS.LOG file for errors and warnings. For more information on supported logging levels and their differences see Configuring VoMS Server topic.

APPENDIX A, Starting VoMS server on Computer Startup

To start VoMS each time Windows starts:

1. Click the Windows **Start** button, and then point to the **Settings**.
2. Click **Taskbar**, and then click the Start Menu **Programs** tab.
3. Click **Add**, and then click **Browse**.
4. Locate VOMS.EXE in the VoMS installation directory, then double-click it.
5. Click **Next**, and then double-click the **StartUp** folder.
6. Type the name **VoMS,** which you will see on the **StartUp** menu, and then click **Finish**. Windows will create a shortcut that will be placed to the **SratUp** folder.
7. Repeat steps 1 and 2. Click **Advanced**, and then locate the newly created shortcut.
8. Right-click on the shortcut, and then click **Properties**
9. For the startup window property select **Minimized**, and then click **OK**

Tips:

When VoMS will start, it will appear as an icon in the Windows System Tray.

To show the VoMS Server Console, double-click on the icon or right-click then select **Show** command from the pop-up menu.

APPENDIX B, Running VoMS server as a Windows NT service

VoMS server can be optionally set to run as a Windows NT service. There are several important Windows NT service features that you should know and carefully consider before setting the VoMS server to run as a service:

The VoMS server can start automatically whenever the computer is started or user is logged to the network and runs continuously in the background.

Use **VOMS.LOG** file to check the server status and activity. This file is located in the same folder as the VoMS programs – by default it is located in the 'C:\Program Files\Voice Message Services' folder.

Tip:
The VoMS server service is not installed and configured automatically on installation. Use `Install/Uninstall VoMS Windows NT Service` shortcuts in the Voice Message Services program group for installing and configuring the VoMS server service.

By default the service is installed under LocalSystem account. You should use Control Panel/Services applet to change the VoMS service account to some other administrative account that has sufficient privileges to access the network. **This limitation is due to Windows NT design, for security purposes. Services running under LocalSystem are started before the system is logged to the network and so they do not have network access.**

For more information on Windows NT services, see your Windows NT documentation. You may also want to visit Microsoft technical support on the Web. The following Microsoft knowledge base articles will be useful:

Q124184 - Service Running as System Account Fails Accessing Network.

Q132679 - Local System Account and Null Sessions in Windows NT.

Q158825 - System and User Account Difference

APPENDIX C, Hardware and Software Requirements

VoMS software requires the following minimum hardware and software configurations:

Minimum Hardware Requirements

Front-end:

1. Intel-based or compatible computer
2. At least 64 MB RAM
3. 12 MB disk space, 18 if SAPI needs to be installed.
4. Voice-modem or Intel Dialog telephony board.
5. VGA monitor

Recommended Configuration

1. Pentium class CPU 400 MHz or better
2. 128 MB RAM or better
3. 18 MB disk space
4. SVGA 256-color or better monitor

Minimum Software Requirements

Back-end:

1. Windows 98 or later
2. TAPI 2.0 or better
3. Modem communication driver

Front-end:

1. Windows server or workstation running one of the following operating system:
   - Windows 2003 (server or workstation)
   - Windows XP (server or workstation)
   - Windows 2000 (server or workstation)
   - Windows NT 4.0 (server or workstation with SP4 or better)
   - Windows Me
   - Windows 98

APPENDIX D, Technical Support

Your questions, comments, and suggestions are welcome.

For technical support email to support@softtreetech.com or use the on-line support form at http://www.softtreetech.com/Support.htm.

Please use the Technical Support Request Form show below when contacting us by email or fax.

When reporting problems, please provide as much information as possible about your problem. Be sure to include the following information:

1 Is the problem reproducible? If so, how?

2 What version of Windows are you running? For example, Windows XP, Windows NT 4.0, etc.

3 What versions of the VoMS server and client software are you running?

4 If a dialog box with an error message was displayed, please include the full text of the dialog box, including the text in the title bar.

5 If the problem involves an external program, provide as much information as possible on this program?

Also, make sure to include the serial number for your copy of the VoMS server or client. Use **Help/About** menu to look up the correct numbers. Registered users have priority support.

For registration information, purchasing or other sales information, please contact our sales department sales@softtreetech.com .

For general information, software updates, the latest information on known problems, and answers to frequently asked questions visit the VoMS home page on the Web: http://www.softtreetech.com/voms/.

We’re happy to help in any way we can, but if you’re having problems please check the troubleshooting section first to see if your question is answered there.

Technical Support Request Form

FAX TO: +1 212-208-4625

EMAIL TO: support@softtreetech.com

Please include the following contact information:

Name____

Company____

Address____

City, State/Zip____

Phone____Fax__________________________________

Email____

Best time to reach you____

Which operating system are you using: MS Windows 9x____   MS Windows NT/2000/XP____

VoMS server and client software versions: server _____   client ______

Computer brand and CPU
type:____

RAM (MB):_____________________________   Speed (Mhz):________________________________

CPU quantity and type: ____

Video driver:____

Relevant devices or
peripherals:____

Description of the problem:

____

____

____

____

____

What steps have you taken in order to solve the problem:

____

____

____

APPENDIX E, Licensing

VOMS SOFTWARE LICENSE

The SOFTWARE PRODUCT is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The SOFTWARE PRODUCT is licensed, not sold.

CAUTION: Loading this software onto a computer indicates your acceptance of the following terms. Please read them carefully.

GRANT OF LICENSE: SoftTree Technologies, Inc. ("SoftTree Technologies") grants you a **limited** license to use the software ("Software").

You may make copies of the Software for backup and archival purposes only. You may permanently transfer all of your rights under this Software LICENSE only in conjunction with a permanent transfer of your validly licensed copy of the product(s).

LICENSE TYPES: The Software and associated add-in components are licensed on a RUN-TIME basis, which means, that for each computer on which the Software is installed, a valid run-time license must exist.

Server License

Permits installation and execution of the "Server" part of the Software on a single computer (a stand-alone computer or a single workstation in a network or a single network server) per license.

You may have up to the number of concurrent users for which you have purchased licenses logged onto the "Server" at any time. Additional licenses must be purchased to install the "Server" Software onto additional computers, at other sites within your organization, or to increase the number of concurrent users.

Client License

Permits installation and execution of the "Client" part of the Software on a single computer (a stand-alone computer or a single workstation in a network or a single network server) per license.

Additional licenses must be purchased to install the "Client" Software onto additional computers.

Site License

Permits installation and execution of the Software on multiple computers within a single physical location (i.e. an office or data center location at a single physical address).

Enterprise License

Permits installation and execution of the Software on multiple computers in multiple locations throughout the licensed company's facilities.

RESTRICTIONS: Unregistered versions (shareware licensed copies) of the Software may be used for a period of not more than 30 days. After 30 days, you must either stop using the Software, or purchase a validly licensed copy.

You must maintain all copyright notices on all copies of the Software. You may not sell copies of the Software to third parties without express written consent of SoftTree Technologies and under SoftTree Technologies' instruction.

EVALUATION copies may be distributed freely without charge so long as the Software remains whole including but not limited to existing copyright notices, installation and setup utilities, help files, licensing agreement, In executing such an act as distributing without the similar copyright or license violation, to the maximum extent permitted by applicable law you may be held liable for loss of revenue to SoftTree Technologies or SoftTree Technologies' representatives due to loss of sales or devaluation of the Software or both.

You must comply with all applicable laws regarding the use of the Software.

COPYRIGHT: The Software is the proprietary product of SoftTree Technologies and is protected by copyright law. You acquire only the right to use the Software and do not acquire any rights of ownership.

For your convenience, SoftTree Technologies provides certain Software components in the source code format. You may customize this code for your environment, but you agree not to publish, transfer, or redistribute in any other form both the original code and the modified code.

You agree not to remove any product identification, copyright notices, or other notices or proprietary restrictions from the Software.

You agree not to cause or permit the reverse engineering, disassembly, or decompilation of the Software. You shall not disclose the results of any benchmark tests of the Software to any third party without SoftTree Technologies' prior written approval.

DESCRIPTION OF OTHER RIGHTS AND LIMITATIONS: You may not rent, lease or transfer the Software except as outlined under GRANT OF LICENSE - use and copy.

Without prejudice to any other rights, SoftTree Technologies may terminate this Software LICENSE if you fail to comply with the terms and conditions of this Software LICENSE. In such event, you must destroy all copies of the Software and all of its component parts.

WARRANTY DISCLAIMER: SoftTree Technologies is providing this license on an "as is" basis without warranty of any kind; SoftTree Technologies disclaims all express and implied warranties, including the implied warranties of merchantability or fitness for a particular purpose.

LIMITATION OF LIABILITY: SoftTree Technologies shall not be liable for any damages, including direct, indirect, incidental, special or consequential damages, or damages for loss of profits, revenue, data or data use, incurred by you or any third party, whether in an action in contract or tort, even if you or any other person has been advised of the possibility of such damages.

SoftTree Technologies, Inc.
Ilyce Ct 62,
Staten Island NY, 10306
USA